Otto-von-Guericke University Magdeburg

Faculty of Computer Science

Master's Thesis

German Occupation Coding Using RAG and LLM-Based Query Rewriting

Author:

Andy Koch

March 21, 2026

Advisors:

Prof. Dr. rer. nat. habil. Gunter Saake
Department of Databases & Software Engineering
Otto-von-Guericke University Magdeburg

Dr.-Ing. David Broneske
Department Infrastructure and Methods
DZHW - German Centre for Higher Education Research & Science Studies

Dr. Andreas Daniel
Department Infrastructure and Methods
DZHW - German Centre for Higher Education Research & Science Studies

Koch, Andy:
German Occupation Coding Using RAG and LLM-Based Query Rewriting
Master's Thesis, Otto-von-Guericke University Magdeburg, 2026.

Abstract

Occupation coding—assigning standardized codes to free-text job descriptions—is central to labor-market and higher-education research. In Germany, this task relies on the five-level Classification of Occupations (KldB 2010) taxonomy; when applied to survey data, the semantic mismatch between colloquial free-text responses and formal taxonomic descriptions makes manual coding expensive, fatigue-prone, and only fair-to-moderately reliable at the finest classification level. Fine-tuned language models achieve strong performance yet depend on labeled training data that is costly to produce and transfers poorly across survey datasets. Recent advances in large language models (LLMs) and retrieval-augmented generation (RAG) offer a label-free alternative, yet no empirical baselines exist for this approach applied to KldB 2010 coding from German survey data under consumer-grade hardware constraints. We implement and evaluate two RAG systems: Naive-RAG, a retrieve-then-classify pipeline, and QR-RAG, which prepends a three-stage LLM-based query-rewriting step. Experiments cover nine embedding models, six LLM query rewriters, and four stratified DZHW graduate-survey datasets on a dual-GPU workstation. Three findings emerge. First, embedding models cluster into three well-separated performance tiers; only the top tier, led by Qwen3-Embedding, provides sufficient retrieval coverage for deployment. Second, query rewriting compensates for weaker embedding models and neutralizes input-complexity variation across datasets, with hypothetical job-task generation as the dominant retrieval signal and Qwen3-30B-MoE as the optimal query rewriter. Third, classification performance reveals a sharp knee point at the 3-to-4-digit transition: at the finest level, both systems yield Cohen's Kappa of 0.26–0.31 despite Recall@20 exceeding 0.73—identifying the classifier, not the retriever, as the binding bottleneck—while at broader 1–3-digit levels both reach Cohen's Kappa of 0.59–0.77, matching the fine-tuned GPT-3 baseline. These results establish the first empirical benchmarks for unsupervised RAG-based KldB 2010 occupation coding under consumer-grade hardware constraints, deliver deployment guidance for resource-constrained institutions, and identify classification improvement as the critical path to further gains.

Acknowledgments

I am grateful to Prof. Dr. Gunter Saake for accepting this thesis at his Department of Databases and Software Engineering and for providing the opportunity to conduct this research.

I thank the DZHW for providing the research topic, real-world datasets, and institutional resources that enabled this work, including Dr. Andreas Daniel's service on the examination committee.

My deepest gratitude goes to Dr.-Ing. David Broneske for his exceptional mentorship throughout this research. Our weekly discussions were invaluable—his thought-provoking questions consistently helped me develop new perspectives, refine my methodology, and strengthen my scientific approach. His encouragement and generous investment of time were particularly instrumental during my transition back to academic work after seven years in vocational practice, and his steadfast support during the project's most challenging phases was deeply appreciated.

Finally, I am grateful to my family and friends for their patience and support during the intensive phases of this thesis, which often demanded my complete focus and attention.

Contents

List of Figures xiii

List of Tables xv

List of Code Listings xvii

List of Acronyms xix

1 Introduction 1
- 1.1 Background and Context . . . 2
  - 1.1.1 The German KldB 2010 Taxonomy . . . 2
  - 1.1.2 Survey-Based Occupation Coding: The Semantic Gap . . . 3
  - 1.1.3 Evolution of Automated Occupation Coding . . . 4
  - 1.1.4 Emerging Technological Opportunities . . . 6
- 1.2 Problem Statement . . . 7
- 1.3 Research Aims and Questions . . . 8
  - 1.3.1 Research Objectives and Aims . . . 9
  - 1.3.2 Investigated RAG Approaches . . . 9
  - 1.3.3 Operationalizing Information Richness . . . 10
  - 1.3.4 Research-Question Hierarchy . . . 10
- 1.4 Structure of the Thesis . . . 12

2 Background 15
- 2.1 Pre-Trained Language Models and Transformers . . . 15
  - 2.1.1 The Shift to Learning-Based Language Processing . . . 15
  - 2.1.2 The Transformer Architecture . . . 16
    - 2.1.2.1 Tokenization and Word Embeddings . . . 16
    - 2.1.2.2 Self-Attention: How Words Inform Each Other . . . 17
    - 2.1.2.3 Multi-Layer Perceptrons: Where Knowledge Lives . . . 18
    - 2.1.2.4 Stacking Layers and the Flow of Meaning . . . 18
  - 2.1.3 Encoder and Decoder Architectures . . . 19
  - 2.1.4 Large Language Models at Scale . . . 19
- 2.2 Text Embedding Models . . . 21
  - 2.2.1 From Static to Contextual Text Representations . . . 21
  - 2.2.2 Semantic Geometry and Cosine Similarity . . . 21
  - 2.2.3 Dual-Encoder Architecture for Efficient Retrieval . . . 22
  - 2.2.4 Multilingual and Instruction-Finetuned Embeddings . . . 22
- 2.3 Retrieval-Augmented Generation . . . 23

2.3.1 Motivation: Limits of Purely Parametric Knowledge 23
2.3.2 The RAG Pipeline 23
2.3.3 LLM-Based Query Rewriting 24
2.4 Summary 24

3 DZHW Survey Data: Quality Assessment, Preprocessing, Dataset Construction 27
3.1 Raw Survey Dataset 28
3.1.1 Origins and Survey Context 28
3.1.2 Structure and Field Composition 28
3.1.3 Surface-Level Data-Quality Issues 29
3.2 Data-Quality Assessment Framework 29
3.2.1 Issue-Origin Dimension 30
3.2.2 Rule-Based Preprocessing Feasibility Dimension 30
3.2.3 LLM-Addressability Dimension 31
3.3 Data-Quality Issues and Classification 31
3.3.1 Preprocessing-Addressable Issues 31
3.3.1.1 Non-Informative Participant Responses 31
3.3.1.2 Structural Data Corruption 32
3.3.1.3 Inconsistent Code Formats 32
3.3.1.4 Invalid Taxonomy Codes 32
3.3.1.5 Systematic Coding Artifacts 33
3.3.2 LLM-Addressable Preserved Issues 33
3.3.2.1 Expression Variability 33
3.3.2.2 Misplaced Task Descriptions 33
3.3.2.3 Unknown Abbreviations 34
3.3.2.4 Inconsistent Capitalization Patterns 34
3.3.2.5 Character-Encoding Degradation 34
3.3.2.6 Semantic Duplicates 35
3.3.3 Inherent Dataset Characteristics 35
3.3.3.1 Multi-Role Entries 35
3.3.3.2 Multi-Label Classification Patterns 36
3.3.4 Issue Classification Overview 36
3.4 Rule-Based Preprocessing Pipeline 38
3.5 Preprocessed Dataset Analysis 40
3.5.1 Class Imbalance and Distribution Patterns 40
3.5.2 Taxonomy Coverage and Code Concentration 41
3.5.3 Multi-Label Classification Prevalence 42
3.5.4 Dataset Complexity and Remaining Challenges 42
3.6 Evaluation Datasets 43
3.6.1 Construction Methodology 44
3.6.1.1 Constraints and Design Rationale 44
3.6.1.2 Input-Complexity Proxy and Stratification Criteria . . 45
3.6.1.3 Stratification Outcomes and Dataset Definitions . . . 45
3.6.1.4 Validity Considerations 46
3.6.2 Dataset Characteristics and Suitability Assessment 46
3.6.2.1 Input-Complexity Analysis 46

3.6.2.2 Preservation of Quantifiable Data-Quality Issues . . . 48
3.6.2.3 Implications of Non-Quantifiable Data-Quality Issues . 50
3.7 Summary and Contributions . . . 52

4 System Implementation 53
4.1 System Design Overview . . . 53
4.1.1 RAG-Paradigm Selection . . . 53
4.1.2 Pipeline-Design Overview . . . 54
4.1.3 Computational Constraints and Implementation Decisions . . . 55
4.2 Knowledge-Base Construction . . . 57
4.2.1 Source Materials . . . 57
4.2.2 Field Extraction and Document Construction . . . 58
4.2.2.1 Field-Selection Rationale . . . 58
4.2.2.2 Task-Description Extraction . . . 60
4.2.2.3 Document Structure . . . 61
4.2.3 Vector-Database Implementation . . . 61
4.2.3.1 Vector-Store Selection . . . 61
4.2.3.2 Embedding-Model Configuration . . . 62
4.2.3.3 Document Indexing Strategy . . . 63
4.3 Shared Pipeline Components . . . 64
4.3.1 Retrieval Stage . . . 64
4.3.2 Classification Stage . . . 65
4.4 Naive-RAG Implementation . . . 67
4.5 QR-RAG Implementation . . . 69
4.5.1 Query-Rewriting Pipeline . . . 70
4.5.1.1 QR Stage 1: Content Refinement . . . 70
4.5.1.2 QR Stage 2: Occupational Attribute Extraction . . . 71
4.5.1.3 QR Stage 3: Hypothetical Task Generation . . . 72
4.5.2 Concurrent Batch-Prompting Architecture . . . 73
4.5.3 Query Formulations . . . 77
4.6 Summary . . . 79

5 Evaluation 83
5.1 Experimental Design . . . 84
5.1.1 Evaluation Datasets . . . 84
5.1.2 Retrieval Knowledge Base . . . 85
5.1.3 Retrieval and Classification Metrics . . . 85
5.1.4 Embedding Models . . . 85
5.1.5 Query-Rewriting LLMs . . . 87
5.1.6 Query Formulations . . . 89
5.2 SQ-1: Naive-RAG Retrieval Performance . . . 90
5.2.1 Performance Tiers Reveal Only Top-Tier EMs Are Viable . . . 91
5.2.2 Job Titles Provide the Dominant Retrieval Signal . . . 94
5.2.3 Long-Tail Gains Establish k = 20 as Optimal Depth . . . 95
5.3 SQ-2: QR-RAG Retrieval Performance . . . 97
5.3.1 Qwen3-30B-MoE Bypasses the Quality–Efficiency Trade-off . . . 100
5.3.2 Hypothetical Tasks Provide the Dominant Retrieval Signal . . . 101
5.4 SQ-3: System-Level Retrieval Performance . . . 102

5.4.1 LLM-Based QR Compensates for Weak Embedding Models . . . 103
5.4.2 LLM-Based QR Neutralizes Input-Complexity Effects 106
5.5 SQ-3: System-Level Classification Performance 108
5.5.1 3-to-4-Digit Transition Reveals Performance Knee Point 109
5.5.2 LLM-Based Classifier Neutralizes Query-Quality Effects 112
5.5.3 Low Conditional Accuracy Despite Strong Retrieval 113
5.5.4 Unsupervised RAG Matches Supervised at Coarse Levels 114
5.6 Summary 116

6 Conclusion 117
6.1 Research Gaps and Aims 118
6.2 Main Results and Contributions 118
6.3 Thesis Organization 120
6.4 Implications 121
6.4.1 Deployment Guidance for Practitioners 121
6.4.1.1 System-Architecture Selection 122
6.4.1.2 Hierarchy-Level Deployment Viability 122
6.4.2 Architectural Insights for RAG-System Designers 122
6.4.2.1 Classification-Bottleneck Localization 122
6.4.2.2 Query-Rewriting Dual Mechanisms 123
6.4.2.3 Hierarchy-Specific Discriminability Limits 123
6.4.3 Methodological Guidance for Researchers 123
6.4.3.1 Data-Quality Assessment Framework 123
6.4.3.2 Stratified Dataset Construction 123
6.4.3.3 Information-Richness Evaluation Design 124
6.5 Limitations 124
6.5.1 External Validity 124
6.5.1.1 Cross-Taxonomy Generalizability 125
6.5.1.2 Cross-Lingual Generalizability 125
6.5.1.3 Cross-Population Generalizability 125
6.5.1.4 Cross-Task Generalizability 125
6.5.2 Internal Validity 126
6.5.2.1 Incomplete Pipeline Ablation 126
6.5.2.2 Dataset-Scale Constraints 126
6.5.2.3 Single-Run Deterministic Execution 126
6.6 Future Work 126
6.6.1 Addressing the Classification Bottleneck 127
6.6.1.1 Hierarchical Classification with Progressive Refinement 127
6.6.1.2 Separate Classification of Orthogonal KldB Dimensions 128
6.6.1.3 Reasoning-Enabled Classification 128
6.6.1.4 Rank-Aware Classification Prompting 129
6.6.1.5 Combinatorial Classification Strategies 129
6.6.2 Complementary Retrieval Optimizations 130
6.6.2.1 Hybrid Dense–Sparse Retrieval 130
6.6.2.2 Combined Retrieval Signals in QR-RAG 130
6.6.2.3 Extended Knowledge Base with Multi-Source Retrieval 131
6.6.2.4 Retrieval Depth Expansion with Reranking 131

6.6.2.5 Integration with Classification Improvements 132
6.7 Open Research Questions 132
6.7.1 Embedding-Model Optimization and Deployment 133
6.7.1.1 Task-Instruction Tuning 133
6.7.1.2 Embedding-Based Classification Alternatives 133
6.7.1.3 Latency Profiling and Model Scaling 133
6.7.2 Reasoning Capabilities Beyond Classification 133
6.7.3 Prompt Engineering and Content Optimization 134
6.7.3.1 Multi-Perspective Task Generation 134
6.7.3.2 Classification-Prompt Enrichment 134
6.7.4 Knowledge-Base Design Alternatives 134
6.7.4.1 LLM-Generated Position-Specific Task Descriptions . . 135
6.7.4.2 Survey-Response Documents 135
6.7.4.3 Integrated Multi-Content KldB Documents 135
6.7.5 Synthetic Benchmark Construction 135
6.7.5.1 High-Quality Ceiling Benchmarks 136
6.7.5.2 Controlled-Degradation Robustness Benchmarks . . . 136

Appendix **137**
A.1 Disclosure of Generative AI Usage 137
A.1.1 AI Systems and Their Applications 137
A.1.2 Quality Assurance and Verification 138
A.1.3 Limitations and Boundaries 138
A.2 Computational Environment 139
A.2.1 Hardware Configuration 139
A.2.2 Software Environment 140
A.2.3 LLM-Inference System 141
A.3 Query-Rewriting Prompt Templates 142
A.4 Evaluation Results 151
A.4.1 SQ-1: Naive-RAG Retrieval Performance 151
A.4.2 SQ-2: QR-RAG Retrieval Performance 151

Bibliography **155**

List of Figures

2.1 Geometric Relationships in Semantic Word-Embedding Space 17
2.2 Original Transformer Model Architecture 20
2.3 Conceptual Framework of a RAG Pipeline 24

3.1 KldB 2010 Code Distribution in the Preprocessed DZHW Dataset . . 41
3.2 Job-Title Length Distributions Across the Four Evaluation Datasets . 47

4.1 Pipeline-Design Overview: Naive-RAG and QR-RAG 56
4.2 Standardized Structure of KldB 2010 Documents 59
4.3 Concurrent Batch-Prompting Architecture 77

5.1 Impact of Embedding Models on Recall Performance in Naive-RAG . 92
5.2 Impact of Embedding Models on NDCG Performance in Naive-RAG . 93
5.3 Impact of QR-LLMs on Recall Performance in QR-RAG 98
5.4 Impact of QR-LLMs on NDCG Performance in QR-RAG 99
5.5 RAG-QF Comparison Across Embedding Tiers: Recall@20 104
5.6 RAG-QF Comparison Across Embedding Tiers: NDCG@20 105
5.7 Accuracy Performance Across the KldB 2010 Hierarchy 110
5.8 Cohen's Kappa Performance Across the KldB 2010 Hierarchy 111

A.1 Custom Workstation for LLM Inference 140

List of Tables

1.1 Hierarchical Structure of the KldB 2010 (v2020) 3

3.1 Raw DZHW Survey Dataset Structure and Representative Examples 29
3.2 Preprocessing-Addressable Issues in the Raw DZHW Survey Dataset 37
3.3 LLM-Addressable Preserved Issues in the Raw DZHW Survey Dataset 37
3.4 Inherent Dataset Characteristics in the Raw DZHW Survey Dataset . 38
3.5 6-Step Rule-Based Preprocessing Pipeline for the DZHW Survey Data 40
3.6 KldB 2010 Code Coverage, Frequencies in the Preprocessed DZHW Dataset 42
3.7 KldB 2010 Code Coverage Across the Four Evaluation Datasets . . . 46
3.8 Job-Title Length Statistics Across the Four Evaluation Datasets . . . 48
3.9 Job-Task Completion Rates Across the Four Evaluation Datasets . . 48
3.10 Encoding, Capitalization Issues Across the Four Evaluation Datasets 50

5.1 Specifications of Selected Multilingual Text Embedding Models 86
5.2 Execution Parameters of Selected Multilingual Text Embedding Models 87
5.3 Specifications of Selected Query-Rewriting LLMs 88
5.4 Execution Parameters of Selected Query-Rewriting LLMs 89
5.5 Selected Query Formulations from Naive-RAG and QR-RAG 90
5.6 Computational Cost of the Query-Rewriting Pipeline in QR-RAG . . 97
5.7 Embedding-Model Performance Tier Composition 103

A.1 Hardware Specifications of the Experimental Workstation 140
A.2 Software Environment Specifications 141
A.3 Ollama Configuration Parameters 142
A.4 Impact of Embedding Models on Recall Performance in Naive-RAG – Numerical Data 152

A.5 Impact of Embedding Models on NDCG Performance in Naive-RAG – Numerical Data 153

A.6 Impact of QR-LLMs on Retrieval Performance in QR-RAG – Numerical Data 154

List of Code Listings

4.1 Template for Knowledge-Base Documents 61
4.2 Knowledge-Base Document for Code 43414 62
4.3 Knowledge-Base Documents with Instruction Prefixes 63
4.4 Classification Prompt Template 66
4.5 Candidate Formatting Template for Classification 66
4.6 Classification Query Injection Across Both Approaches 67
4.7 Classification Prompt Example 68
4.8 Naive-RAG Query Formulation Templates 69
4.9 QR Stage 1 Input–Output Schemas 71
4.10 QR Stage 1 Transformation Example 72
4.11 QR Stage 2 Input–Output Schemas 73
4.12 QR Stage 2 Transformation Example 74
4.13 QR Stage 3 Input–Output Schemas 75
4.14 QR Stage 3 Transformation Example 76
4.15 Shared 7-Step LangChain Runnable Sequence 78
4.16 QR-RAG Query Formulation Templates 81
4.17 QR-RAG Query Formulation Examples 82
A.1 QR Stage 1 Prompt Template: Content Refinement 143
A.2 QR Stage 2 Prompt Template: Occupational Attribute Extraction . . 145
A.3 QR Stage 3 Prompt Template: Hypothetical Task Generation 149

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
BA	German Federal Employment Agency
BERT	Bidirectional Encoder Representations from Transformers
BF16	16-Bit Brain Floating-Point
BM25	Best Matching 25
CASCOT	Computer Assisted Structured COding Tool
CSV	Comma-Separated Values
DKZ	Digital Classification Center
DQAF	Data-Quality Assessment Framework
DZHW	German Centre for Higher Education Research and Science Studies
EM	Embedding Model
ESCO	European Skills, Competences, Qualifications and Occupations
FP16	16-Bit Floating-Point
GELU	Gaussian Error Linear Units
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
HNSW	Hierarchical Navigable Small World
IR	Information Retrieval
ISCO	International Standard Classification of Occupations
JSON	JavaScript Object Notation
KldB	Classification of Occupations
kNN	k-Nearest Neighbors
LJT	Longest Job Title Dataset
LLM	Large Language Model
LoRA	Low-Rank Adaptation
ML	Machine Learning
MLP	Multi-Layer Perceptron
MoE	Mixture-of-Experts
MRQ	Main Research Question
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
O*NET	Occupational Information Network
PDF	Portable Document Format
PLM	Pre-trained Language Model
PPD	Preprocessed Dataset
QF	Query Formulation

QR	Query Rewriting
RAD	Random All Dataset
RAG	Retrieval-Augmented Generation
ReLU	Rectified Linear Unit
RVD	Random Valid Dataset
SJT	Shortest Job Title Dataset
SOC	Standard Occupational Classification
SQ	Subquestion
VRAM	Video Random Access Memory
XGBoost	Extreme Gradient Boosting

1. Introduction

Occupation coding assigns standardized codes to natural-language job descriptions [Schierholz, 2014] and underpins labor-market research [Johary et al., 2025; Rony and Patman, 2025] and epidemiological studies [Langezaal et al., 2023; Russ et al., 2016]. Research institutions like the German Centre for Higher Education Research and Science Studies (DZHW)[1] conduct large-scale surveys [Fabian et al., 2023; Hoffstätter et al., 2021] that demand manual[2] occupation coding: they analyze academic career trajectories, assess skill mismatches between higher education and employment, and track how graduates transition across occupations over time [Fabian et al., 2021]. In Germany, such coding relies on the Classification of Occupations (KldB) 2010, the country's authoritative occupation taxonomy organizing 1,300 categories across five hierarchical levels [Bundesagentur für Arbeit, 2021a,b; Paulus and Matthes, 2013].

Expert coders manually assign occupation codes across diverse data sources, but survey-based occupation coding presents distinctive challenges because surveys typically collect occupational data as free-text responses to open-ended questions. The semantic gap between informal respondent descriptions and formal taxonomic job categories drives these difficulties [Schierholz, 2014]. High costs and processing bottlenecks compound the problem [Langezaal et al., 2025; Patel et al., 2012]. Recent advances in retrieval-augmented generation (RAG) and pre-trained language models (PLMs)—including transformer-based architectures like BERT and GPT—now offer new opportunities for automated and computer-assisted coding systems. Yet their effectiveness for German survey-based occupation coding remains empirically uncharacterized.

This chapter establishes the research context by addressing four key areas. We begin by examining occupation-coding fundamentals and current practices (Section 1.1), then identify the specific research problem (Section 1.2). After defining our research objectives and questions (Section 1.3), we conclude by outlining how the remaining chapters unfold (Section 1.4).

[1] https://www.dzhw.eu/en/gmbh/index_html
[2] from personal communication

1.1 Background and Context

The KldB 2010 taxonomy structures occupational knowledge across 1,300 categories, serving labor-market researchers, epidemiologists, and research institutions like the DZHW. Organizations source employment data from résumés, job advertisements, and surveys, yet these sources present distinct challenges. Survey responses exhibit particular difficulties: respondents use variable, informal descriptions that diverge from standardized taxonomic terminology. While manual coding remains standard practice, the semantic gap between colloquial language and formal categories creates interpretation challenges that resource constraints and processing demands amplify. These challenges motivate automated approaches using PLMs and RAG architectures, though their effectiveness for German survey-based occupation coding lacks empirical characterization. This section examines four dimensions: the KldB taxonomy structure and its application contexts (Section 1.1.1), the semantic gap in survey-based occupation coding (Section 1.1.2), the evolution of automated occupation coding (Section 1.1.3), and emerging technological opportunities (Section 1.1.4).

1.1.1 The German KldB 2010 Taxonomy

The KldB 2010, in its revised 2020 version, organizes 1,300 occupational categories into a 5-level hierarchy, ranging from broad occupational areas to specific occupational types (Table 1.1). Its 2-dimensional structure classifies occupations along a horizontal dimension of occupational specialization (based on thematic fields and specific task profiles) and a vertical dimension of requirement levels (based on activity complexity and necessary formal qualifications). The horizontal dimension structures the first four hierarchical levels—from occupational areas (10 categories) through occupational main groups (37 categories), occupational groups (144 categories), to occupational subgroups (702 categories)—while the vertical dimension is represented by the 5th digit, distinguishing four requirement levels from unskilled helper activities to complex specialist tasks requiring master's or doctoral degrees [Bundesagentur für Arbeit, 2021a,b]. Unlike international taxonomies such as International Standard Classification of Occupations (ISCO) 2008, which integrates skill levels hierarchically across major occupational groups, or European Skills, Competences, Qualifications and Occupations (ESCO), which maintains separate pillars for occupations and skills, the KldB's two dimensions are empirically derived and orthogonal (via cluster analysis), enabling independent analysis of occupational content and skill complexity [Christoph et al., 2020; Müller, 2024; Paulus and Matthes, 2013].

This structure enables fine-grained labor-market analysis while maintaining statistical aggregation capabilities, yet it also creates three interconnected coding challenges. First, the 2-dimensional architecture requires coders to navigate both occupational specialization and requirement level simultaneously [Paulus and Matthes, 2013; Schierholz et al., 2018]. Second, the hierarchical structure demands decisions about appropriate classification granularity across five levels, with finer-grained distinctions at lower levels requiring more detailed occupational knowledge [Schierholz and Schonlau, 2020]. Third, the mismatch between the KldB's formal German occupational terminology and the colloquial language used in survey responses creates a linguistic barrier, exacerbated by compound-noun structures and domain-specific technical

Table 1.1: Hierarchical Structure of the KldB 2010 (v2020). The KldB 2010 employs a 5-level hierarchy where code length (1–5 digits) corresponds to classification granularity. The first four levels (horizontal dimension) progressively narrow occupational specialization from broad areas to specific subgroups, while the 5th digit (vertical dimension) encodes requirement levels based on task complexity and qualification demands. Category counts are from the 2020 revision.

Level	Code	German Name	English Translation	Categories	Dimension
1	1-digit	Berufsbereiche	Occupational Areas	10	Horizontal
2	2-digits	Berufshauptgruppen	Occupational Main Groups	37	Horizontal
3	3-digits	Berufsgruppen	Occupational Groups	144	Horizontal
4	4-digits	Berufsuntergruppen	Occupational Subgroups	702	Horizontal
5	5-digits	Berufsgattungen	Occupational Types	1,300	Vertical

vocabulary that may not appear in respondents' everyday occupational descriptions [Dorau and Hein, 2025; Schierholz, 2014; Schierholz et al., 2018].

The taxonomy serves diverse research and policy communities. Labor-market researchers use KldB codes to analyze wage differentials, occupational mobility, and employment trends [Bayer and Kuhn, 2023; Fischer and Dörpinghaus, 2024; Rickmeier, 2023]. Epidemiologists examine occupation–disease relationships and occupational health outcomes [Eyles et al., 2019; Reuter et al., 2022]. Research institutions such as DZHW maintain extensive occupation-coding operations for higher education and career-trajectory research, manually processing diverse occupational descriptions from graduates, researchers, and academic professionals [Fabian et al., 2023; Hoffstätter et al., 2021] to understand skill utilization, career progression, and the alignment between education and employment [Fabian et al., 2021]. As artificial intelligence (AI) and demographic shifts reshape the German labor market, creating new occupations while rendering others obsolete [Dengler and Matthes, 2019; Warning et al., 2022], the need for efficient and reliable occupation coding has intensified.

1.1.2 Survey-Based Occupation Coding: The Semantic Gap

While occupational data from résumés, job advertisements, and surveys each require different coding approaches, survey data poses a fundamental challenge: bridging a *semantic gap*, the difference in meaning between constructs formed within different representation systems [Hein, 2010]. This semantic gap manifests in two dimensions. A *register mismatch* separates colloquial survey responses from formal KldB definitions. Respondents describe work using contextual language; the KldB employs standardized terminology [Schierholz, 2014]. A *granularity mismatch* separates position-oriented job descriptions from category-oriented classifications. Respondents describe specific positions with concrete details; the KldB defines broader categories that abstract from workplace contexts. These mismatches create interpretation challenges that affect coding accuracy and efficiency—challenges that systematic characteristics of survey responses themselves compound.

Two characteristics of survey responses impede unambiguous classification. *Terminological diversity* creates substantial variation: even within identical occupational

categories, job name descriptions average only 0.23 cosine similarity, while task descriptions average 0.10 [Martin-Caughey, 2021]. Respondents employ organization-specific titles, domain jargon, and colloquial terms that diverge from formal KldB labels [Schierholz, 2014]. *Information density* varies dramatically. Some responses provide minimal detail—generic titles like "clerk" [Schierholz, 2014] or vague designations like "employee of department X" [Tijdens, 2014]. Others supply extensive descriptions including tasks, qualifications, hierarchical position, and industry context [Belloni et al., 2016; Kononykhina et al., 2025]. Terminological heterogeneity and variable density converge to produce *ambiguity* in mapping free-text descriptions to standardized categories. Responses lack sufficient specificity to distinguish similar categories, permit multiple plausible classifications, or provide contextual details that obscure categorical membership [Conrad et al., 2016; Schierholz, 2014].

Low inter-coder reliability reveals these ambiguities in practice. Formal taxonomies cannot capture the ambiguity in natural-language descriptions, forcing coders to apply subjective judgment and informal rules [Conrad et al., 2016]. Inter-coder agreement ranges from 50% to 86% across coding levels even among trained experts [Russ et al., 2016], with Cohen's Kappa values of 0.34 to 0.61 at the finest level on French taxonomies [Langezaal et al., 2025]. Error rates reach 28% to 30% even at the broad 1-digit level of ISCO-88 [Belloni et al., 2016]. These disagreement rates persist despite auxiliary variables like industry sector and educational requirements [Belloni et al., 2016].

Resource constraints compound semantic challenges. Expert coding requires specialized training and sustained personnel allocation, creating substantial expenses [Langezaal et al., 2025; Patel et al., 2012; Schierholz et al., 2018]. Coders achieve roughly 2,700 codes per month [Langezaal et al., 2025]—roughly 120 per working day. Prolonged sessions reduce accuracy through fatigue [Langezaal et al., 2023]. Large epidemiological cohorts demand multiple coders, exacerbating inter-coder consistency challenges. Panel studies create continuous coding demands as respondents change occupations.

These challenges drive development of automated and computer-assisted systems that accelerate workflows and reduce costs while maintaining quality [Langezaal et al., 2025; Schierholz et al., 2018; Wan et al., 2023]. For institutions like DZHW conducting continuous survey-based coding [Safikhani et al., 2023], such systems could enable semi-automated workflows: automated systems handle straightforward cases while routing ambiguous cases to expert review.

1.1.3 Evolution of Automated Occupation Coding

Automated occupation coding has historically relied on coding-index and rule-based approaches, ranging from exact matching against standardized job-title lists to similarity-based matching tools such as CASCOT[3] [Gweon et al., 2017; Schierholz and Schonlau, 2020]. Early work (e.g., [O'Reagan, 1972]) already framed rule-based coding as a core automation strategy, and later practice-oriented systems operationalized occupation coding as large dictionaries or rule sets paired with

[3] https://warwick.ac.uk/fac/soc/ier/data_group/cascot/choose_classificatio/

thresholds for partial automation rather than full replacement of human coders [Gweon et al., 2017].

Building on these foundations, research increasingly leverages statistical learning trained on previously coded answers, including kNN-style "memory-based reasoning" [Creecy et al., 1992] and adapted nearest-neighbor approaches that additionally output confidence scores for human review [Schierholz and Schonlau, 2020]. In a systematic comparison across five German survey datasets coded to the KldB 2010, Schierholz and Schonlau [2020] find that tree boosting (XGBoost) becomes most effective when training data are enriched by appending job titles from the official coding index to previously coded survey answers (their "Algorithm 7"), thereby combining coding-index coverage with statistical learning from survey data.

Since late 2018, transformer-based language models [Vaswani et al., 2017] have reshaped natural language processing (NLP) [Min et al., 2023]. Within this paradigm, encoder architectures such as Bidirectional Encoder Representations from Transformers (BERT) support strong semantic text understanding [Devlin et al., 2019], while decoder architectures such as Generative Pre-trained Transformer (GPT) enable fluent text generation [Radford and Narasimhan, 2018]. For German occupation coding, supervised fine-tuning of PLMs has demonstrated substantial potential [Safikhani et al., 2023]. On challenging DZHW survey data evaluated at the finest 5-digit level of the KldB 2010, BERT achieved a Cohen's Kappa of 0.64 [Safikhani et al., 2023]. This result substantially outperformed both fine-tuned GPT-3[4] (0.30) and the traditional machine learning (ML) baseline (0.49)—specifically a state-of-the-art XGBoost ensemble enhanced by dictionary lookups as proposed by Schierholz and Schonlau [2020] [Safikhani et al., 2023].

However, the widespread deployment of supervised occupation-coding models is often constrained by the availability of sufficiently large, high-quality training data, since creating such training labels requires expensive manual coding [Schierholz and Schonlau, 2020]. Furthermore, Schierholz and Schonlau [2020] report that performance differences between algorithms are often smaller than the performance variation observed across datasets, which they attribute to systematic differences in coding practices (e.g., across coding agencies and studies). This limits straightforward cross-survey transfer and risks perpetuating questionable or inconsistent historical coding decisions when models are trained on previously coded data [Schierholz and Schonlau, 2020].

These constraints increase the importance of partial-automation workflows in practice. Consistent with long-standing occupation-coding systems, many approaches rely on confidence scores (or predicted probabilities) to automatically code easy cases while routing low-confidence cases to expert coders, making ranked candidate suggestions and threshold-based decision rules central for human-in-the-loop deployment [Gweon et al., 2017; Schierholz and Schonlau, 2020]. In line with this operational perspective, Safikhani et al. [2023] identify human-in-the-loop workflows as a key future requirement for DZHW survey practice.

Despite these supervised benchmarks, unsupervised large language model (LLM)-based approaches—which leverage PLMs through prompting rather than fine-tuning—

[4] https://github.com/openai/gpt-3

remain largely unexplored for KldB 2010 coding, even as they achieve strong results on international taxonomies. For example, Achananuparp et al. [2025] propose a multi-stage pipeline with inference, retrieval, and reranking, achieving 0.76 to 0.81 Precision@1 on the US-centric O*NET–SOC taxonomy (e.g., 0.7612 for Llama3-8B[5] and 0.8114 for GPT-3.5-Turbo[6]) using taxonomy-guided reasoning. Their evaluation targets the finest 8-digit granularity and uses the pre-filtered "Jobs12K" test set (labeled by GPT-4o[7]), consisting of job-title–company pairs from US-based résumé data [Achananuparp et al., 2025]. Under the same evaluation conditions, this pipeline substantially outperforms the earlier "LLM4Jobs" framework [Li et al., 2025], which reaches lower Precision@1 across several models (e.g., 0.5261 for Llama3-8B and 0.6384 for GPT-3.5-Turbo) [Achananuparp et al., 2025]. This multi-stage pipeline exemplifies a retrieval-augmented generation (RAG) architecture [Lewis et al., 2020] applied to automated occupation coding. The pipeline retrieves taxonomy-relevant documents, reranks them by relevance, and augments LLM prompts with this refined context to generate the final classification.

1.1.4 Emerging Technological Opportunities

Several technological advances create new opportunities for German survey-based occupation coding. First, RAG systems integrate external knowledge into LLM prompts, reducing hallucinations [Shuster et al., 2021] and improving factual accuracy in knowledge-intensive tasks [Lewis et al., 2020]. For occupation coding, RAG architectures enable smaller LLMs—which often lack internal knowledge of specialized occupational taxonomies [Achananuparp et al., 2025]—to perform classification by retrieving relevant taxonomic knowledge at inference time. For instance, Achananuparp et al. [2025] show that efficient LLMs with fewer than 10 billion parameters demonstrate limited internal taxonomic knowledge: on "complete-recall" tasks requiring code generation without external hints, Llama3-8B achieves only 0.11 accuracy, and across all knowledge-assessment tasks averages 0.28 accuracy. Yet when supported by retrieval from the O*NET–SOC taxonomy within a multi-stage RAG framework, the same model reaches 0.76 accuracy for occupation classification. This separation of knowledge (external documents) from LLM inference also enables straightforward knowledge-base updates without model retraining [Lewis et al., 2020], a key advantage when taxonomies evolve or new occupational categories emerge. This retrieval-centered design naturally transitions to the embedding stage, where semantic similarity between occupational data and taxonomic documents is computed using text embeddings.

Second, multilingual text embedding models (EMs) based on dual-encoder architectures enable efficient retrieval in RAG systems by allowing pre-computation of document embeddings and fast semantic similarity matching at query time [Karpukhin et al., 2020]. Recent multilingual text EMs have progressed from BERT [Devlin et al., 2019] to modern architectures like XLM-RoBERTa[8]-based variants, including Multilingual-E5 [Wang et al., 2024b] and Jina-Embeddings-v3 [Sturua et al., 2024],

[5] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[6] https://platform.openai.com/docs/models/gpt-3.5-turbo
[7] https://platform.openai.com/docs/models/gpt-4o
[8] https://huggingface.co/docs/transformers/en/model_doc/xlm-roberta

which achieve strong performance on German-language retrieval tasks while supporting substantially longer context windows (e.g., 8,192 tokens vs. BERT's 512-token limit). Additionally, models like Jina-Embeddings-v3 incorporate task-specific low-rank adaptation (LoRA) modules [Sturua et al., 2024], and instruction-finetuned EMs support task-differentiated encoding through natural-language instructions that specify text type, task objective, and domain [Su et al., 2023], allowing embeddings to be optimized for occupation-coding workflows. Most recently, EMs built directly on LLM foundation models—such as Qwen3-Embedding [Zhang et al., 2025b]—inherit the multilingual text understanding and reasoning capabilities of their underlying LLMs, achieving state-of-the-art performance on multilingual retrieval benchmarks while maintaining the efficiency of dual-encoder architectures.

Finally, LLMs demonstrate strong text-generation and synthetic-data augmentation capabilities [Ding et al., 2024], enabling document generation for knowledge-base enrichment [Hong et al., 2024; Wu and Cao, 2024] and query generation for improved retrieval [Gao et al., 2023; Kim et al., 2024; Koo et al., 2024; Ma et al., 2023; Wang et al., 2023]. For survey-based occupation coding, where response quality and information density vary substantially due to informal language, missing details, and question-framing effects [Kononykhina et al., 2025; Safikhani et al., 2023; Schierholz et al., 2018], LLM-based query rewriting offers a mechanism to enrich terse job descriptions or clarify ambiguous ones before retrieval, potentially improving query–document alignment with formal taxonomic descriptions. Combined with expanded context windows supporting millions of tokens [An et al., 2024; Xiong et al., 2024; Xu et al., 2025], improved in-context learning [Peng et al., 2024; Voronov et al., 2024], and reasoning capabilities [Guo et al., 2025; Patil and Jadon, 2025], these advances enable LLMs to perform retrieval-augmented classification by synthesizing evidence from multiple retrieved documents within a single prompt context. While quantization techniques [Husom et al., 2025], sparsely-activated Mixture-of-Experts (MoE) architectures [Cai et al., 2025; Yang et al., 2025; Zhang et al., 2025a], and local inference frameworks[9] have improved the feasibility of deploying LLMs on consumer-grade hardware, autoregressive token generation remains computationally expensive.

1.2 Problem Statement

Recent work has clarified key challenges of survey-based occupation coding for the KldB 2010 and introduced new methodological opportunities, yet critical questions about effectiveness, transferability, and feasible local deployment remain unanswered. This section specifies the open research gaps that this thesis addresses.

Supervised PLM fine-tuning achieves strong performance for German survey-based occupation coding [Safikhani et al., 2023], but widespread deployment faces constraints: labeled-data requirements and cross-dataset variation in coding practices limit applicability [Schierholz and Schonlau, 2020]. Unsupervised multi-stage LLM pipelines show promise on international taxonomies and résumé datasets [Achananuparp et al., 2025; Li et al., 2025]. However, these pipelines operate under different taxonomy, language, and dataset conditions. They typically rely on proprietary

[9] Such as vLLM (https://docs.vllm.ai/) and Ollama (https://ollama.com/).

frontier models (GPT-4o, GPT-3.5-Turbo) and remote API-based inference, limiting reproducibility and local deployment.

RAG offers a principled approach: retrieve taxonomic evidence at inference time, then condition LLM decisions on retrieved documents. Multilingual text EMs become critical in this architecture because they operationalize semantic similarity between queries and documents during retrieval. For the KldB 2010 and German survey responses specifically, EMs must match free-text survey responses against formal taxonomic documents. Recent RAG research frames LLM-based query rewriting as a pre-retrieval mechanism: transform underspecified or noisy inputs into retrieval-effective queries through rewrite–retrieve–read pipelines [Koo et al., 2024; Ma et al., 2023]. This transformation improves both retrieval quality and downstream performance.

To the best of our knowledge, no empirical baselines exist for unsupervised RAG-based occupation coding on the KldB 2010 from German survey data under local, consumer-grade hardware constraints. No systematic characterization exists of how performance varies with the KldB's 2-dimensional structure across five hierarchy levels (1–5 digits), German-language specificity, and survey-response characteristics in real settings (e.g., DZHW survey data). Key RAG design choices remain empirically uncharacterized: (i) which multilingual text EMs best support retrieval of KldB documents from German free-text survey responses, and (ii) whether LLM-based query rewriting robustly improves performance—and which LLMs prove most suitable—given the additional computational cost of local inference.

This absence creates practical barriers. Smaller survey organizations and research institutions implementing occupation-coding workflows under data-privacy and hardware constraints cannot validate whether chosen systems approach optimality or justify computational and personnel trade-offs. Scientifically, the lack of comparative evaluation obscures when RAG and LLM-based query rewriting provide genuine benefits versus unnecessary computational overhead for German survey-based occupation coding.

This thesis addresses these gaps by establishing empirical baselines for unsupervised RAG-based occupation coding on the KldB 2010 from German survey data under consumer-grade hardware constraints. It evaluates whether LLM-based query rewriting provides measurable improvements. The next section operationalizes these contributions through explicit research aims and a hierarchy of research questions.

1.3 Research Aims and Questions

This thesis pursues two research aims addressing the identified gaps in German occupation coding: establishing empirical baselines for unsupervised RAG-based KldB 2010 occupation coding under consumer-grade hardware constraints, and evaluating whether LLM-based query rewriting improves performance. We operationalize these aims through a main research question (MRQ) supported by three subquestions (SQs). The following subsections articulate these research aims in detail (Section 1.3.1), define the specific RAG approaches under comparison (Section 1.3.2), specify the dimensions of information richness evaluated (Section 1.3.3), and present the formal research-question hierarchy (Section 1.3.4).

1.3.1 Research Objectives and Aims

Building on the identified gaps in German survey-based occupation coding (Section 1.2), we pursue two complementary research aims. The first establishes robust empirical baselines for unsupervised RAG-based occupation coding on the KldB 2010 taxonomy under consumer-grade hardware constraints. The KldB's distinct taxonomy structure and German-language specificity, combined with the unique characteristics of survey-response data and the reliance of existing international approaches on proprietary models and remote inference, preclude direct transfer of methods developed for international taxonomies. We aim to develop and characterize two system implementations: (i) *Naive-RAG*, operationalizing the *standard RAG* approach; and (ii) *QR-RAG*, operationalizing the *query-rewriting-enhanced RAG* approach. Systematically exploring their design spaces provides the first benchmarks for KldB coding under strict resource constraints.

The second aim determines whether LLM-based query rewriting enhances retrieval and classification performance relative to standard RAG. We aim to evaluate this effectiveness across varying information-richness conditions (Section 1.3.3) and across KldB hierarchy levels (Table 1.1), identifying the specific conditions where query transformation provides greatest marginal utility.

Ensuring practical viability constitutes a cross-cutting objective. LLM inference demands substantial computation, yet we want to demonstrate that sophisticated RAG pipelines deploy effectively on consumer-grade hardware (Appendix A.2.1). This provides smaller research institutions and survey organizations with deployment guidance reflecting actual operational and financial limitations, moving beyond theoretical performance to real-world feasibility.

1.3.2 Investigated RAG Approaches

We compare two RAG approaches differing in query-preparation strategies. Our Naive-RAG system implements the standard RAG approach, serving as the primary experimental baseline. It follows a 3-stage pipeline: (1) query formulation structures free-text survey responses into search queries; (2) retrieval selects relevant KldB documents from a knowledge base using semantic similarity; (3) LLM-based classification assigns the most appropriate KldB code based on retrieved documents.

Our QR-RAG system implements the query-rewriting-enhanced RAG approach by introducing LLM-based query rewriting before query formulation. Query-rewriting LLMs transform and enrich German survey responses to improve query–document alignment, increasing downstream retrieval performance. The three pipeline stages (1–3) proceed with these enhanced query representations.

The approaches differ across specific design spaces. Standard RAG performance depends on embedding-model selection and query-formulation choices (which occupational fields to include in search queries). Query-rewriting-enhanced RAG additionally depends on selecting the query-rewriting LLM (which language model transforms the queries). We characterize both approaches across their respective design spaces before comparing them in SQ-3.

1.3.3 Operationalizing Information Richness

To assess whether LLM-based query-rewriting benefits generalize across diverse survey-response characteristics, we evaluate both approaches under varying information-richness conditions. We operationalize information richness through two dimensions. First, *input complexity* evaluates four datasets stratified by job-title length, ranging from minimal to extensive job descriptions. This reflects natural variation in survey-response quality encountered in operational research settings. Second, *context density* manipulates query formulations (QFs) that include varying occupational detail in search queries, from single-field (*Title-Only*, *Tasks-Only*) to multi-field (*Title+Tasks*, *Schema-Data*) queries.

Systematically varying both dimensions independently isolates whether query rewriting (QR) advantages are input-driven or method-driven. This identifies the specific conditions where performance improvements prove most pronounced in real-world operational research settings.

1.3.4 Research-Question Hierarchy

We operationalize our research aims through one main research question supported by three subquestions. The main question addresses whether LLM-based query rewriting improves performance and under which conditions improvements prove most pronounced. The three subquestions follow component-to-system progression: the first two characterize each approach's retrieval performance independently, identifying optimal configurations for standard RAG and query-rewriting-enhanced RAG; the 3rd compares both approaches at the system level under optimal configurations.

This structure operationalizes our first aim (establishing empirical baselines) through the first two subquestions and our second aim (comparatively evaluating QR effectiveness) through the 3rd subquestion, while enabling assessment of how performance varies across information-richness conditions and taxonomic granularity. We present the MRQ followed by the three SQs with their detailed rationales below.

MRQ: LLM-Based QR Effectiveness for RAG-Based Occupation Coding

To what extent does query-rewriting-enhanced RAG outperform standard RAG for unsupervised KldB 2010 occupation coding in retrieval and classification performance, and under which information-richness conditions and hierarchy levels are improvements most pronounced?

SQ 1: Retrieval Baseline for Standard RAG (Naive-RAG)

What empirical baseline can we establish for retrieval performance in unsupervised KldB 2010 occupation coding from survey data using standard RAG approaches across embedding models, query formulations, and retrieval depths; and which embedding model and retrieval depth optimize retrieval effectiveness for downstream experiments?

To establish an empirical baseline for unsupervised KldB occupation coding from survey data using standard RAG approaches, we characterize Naive-RAG's retrieval performance across its design space. Embedding models exhibit substantial performance variation across retrieval tasks [Thakur et al., 2021], yet their effectiveness for retrieving KldB documents using German free-text survey responses remains empirically uncharacterized.

We evaluate Naive-RAG across nine embedding models (monolingual German and multilingual architectures), three query formulations (Title-Only, Tasks-Only, Title+Tasks), and three retrieval depths ($k = 5, 10, 20$), measuring Recall@k and NDCG@k aggregated across all four DZHW evaluation datasets. This establishes the first empirical baseline for standard RAG on survey-based KldB occupation coding while identifying optimal configurations for downstream experiments in SQ-3.

SQ 2: Retrieval Baseline for QR-Enhanced RAG (QR-RAG)

What empirical baseline can we establish for retrieval performance in unsupervised KldB 2010 occupation coding from survey data using query-rewriting-enhanced RAG approaches across query-rewriting LLMs, query formulations, and retrieval depths; and which LLM-based query rewriter optimizes both retrieval effectiveness and computational efficiency for downstream experiments?

To establish an empirical baseline for unsupervised KldB occupation coding from survey data using query-rewriting-enhanced RAG approaches, we characterize QR-RAG's retrieval performance across its design space. LLM-based query rewriting may enhance retrieval by improving query–document alignment through input enrichment and restructuring [Lee, 2025], potentially bridging the semantic gap between informal survey responses and formal KldB documents. LLMs vary substantially in scale and architectural design, yet their effectiveness as query rewriters for this task remains empirically uncharacterized.

We evaluate QR-RAG across six QR-LLMs, four query formulations (Title-Only, Tasks-Only, Title+Tasks, Schema-Data), and three retrieval depths ($k = 5, 10, 20$). Unlike the first subquestion aggregating only over evaluation datasets to isolate EM effects, here we aggregate over both evaluation datasets and embedding models to isolate LLM-based query-rewriting effects. We measure retrieval performance using Recall@k and NDCG@k while assessing computational efficiency through execution times under consumer-grade hardware constraints. This establishes the first empirical baseline for query-rewriting-enhanced RAG on survey-based KldB occupation coding while identifying the QR-LLM that optimizes both retrieval effectiveness and computational efficiency for downstream experiments in SQ-3.

SQ 3: Comparative Evaluation: Retrieval and Classification

Under optimal configurations, how do query-rewriting-enhanced RAG and standard RAG compare: (i) in retrieval performance across information-richness conditions; and (ii) in classification performance across KldB 2010 hierarchy levels?

Having established empirical baselines for both approaches through systematic characterizations in the first two subquestions, this question comparatively evaluates them at the system level using optimal configurations. This comparison determines whether LLM-based query rewriting provides performance advantages over standard RAG for unsupervised KldB occupation coding from survey data and characterizes the conditions where such advantages prove most pronounced.

We evaluate each RAG approach across query formulations and all four DZHW evaluation datasets, preserving information-richness variation to reveal how performance differs with survey-response characteristics. For retrieval comparison, we employ the optimal retrieval depth identified in the first subquestion and confirmed in the second, with query-rewriting-enhanced RAG utilizing the optimal QR-LLM. We measure retrieval performance using Recall@k and NDCG@k across all nine embedding models, stratifying results by EM performance tier to examine whether QR benefits vary systematically across EM capabilities.

For classification comparison, we employ the optimal embedding model and retrieval depth, with the optimal QR-LLM serving dual roles as classifier for both approaches and as query rewriter for query-rewriting-enhanced RAG. We focus on the multi-field QFs enabling unbiased cross-system comparison, computing classification metrics (Cohen's Kappa, Accuracy) across the five KldB hierarchy levels (1-digit to 5-digit codes). Together, these comparisons establish whether query-rewriting-enhanced RAG delivers robust performance gains for survey-based occupation coding and identify the specific information-richness conditions and hierarchy levels where improvements prove most pronounced.

1.4 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 builds the conceptual vocabulary the thesis depends on throughout, covering three interconnected topics: the transformer-based language models that enable semantic understanding, the embedding models that translate language into geometry for retrieval, and the RAG paradigm—its indexing–retrieval–generation pipeline and query-rewriting techniques for bridging the gap between informal survey responses and formal taxonomy entries.

Alongside these technical foundations, Chapter 3 confronts the challenge of creating authentic evaluation data from messy real-world survey responses. It introduces the data-quality assessment framework (DQAF) to distinguish issues that preprocessing can fix from those that should be preserved to maintain evaluation authenticity, implements a rule-based pipeline to resolve the former, and constructs four evaluation datasets through stratified sampling—one maximizing taxonomic breadth, three operationalizing input complexity through controlled job-title selection.

With evaluation data in place, Chapter 4 implements the two RAG architectures under evaluation. It justifies the vector-based RAG paradigm given data-privacy and hardware constraints, constructs the shared knowledge base from all KldB occupational descriptions, and specifies Naive-RAG's direct query-formulation strategies alongside QR-RAG's three-stage query-rewriting pipeline with its concurrent batch-prompting architecture.

Drawing on these datasets and systems, Chapter 5 answers the three subquestions through systematic experimentation. It first isolates embedding-model effects to characterize Naive-RAG's retrieval performance, then isolates query-rewriting effects to characterize QR-RAG's, before conducting system-level comparisons that examine how information richness moderates query-rewriting benefits and how retrieval performance translates into classification accuracy across the five KldB hierarchy levels.

Finally, Chapter 6 synthesizes these findings into six key contributions and translates them into guidance for practitioners, system architects, and researchers. It defines the validity constraints bounding generalizability, proposes targeted improvements for identified bottlenecks, and maps the open research questions whose answers remain empirically uncharacterized.

2. Background

This chapter introduces the technical foundations underlying the two RAG-based occupation-coding systems evaluated in this thesis. We cover three interdependent building blocks: pre-trained language models and the transformer architecture that powers them (Section 2.1), text embedding models that convert language into geometric vectors suitable for semantic search (Section 2.2), and retrieval-augmented generation, the architectural paradigm that integrates both components into a unified pipeline (Section 2.3).

2.1 Pre-Trained Language Models and Transformers

This section provides the theoretical foundation for the NLP techniques we employ in this thesis. We begin by discussing the paradigm shift from rule-based systems to learning-based language processing (Section 2.1.1). We then detail the internal mechanics of the transformer architecture, which underpins modern language models (Section 2.1.2), before distinguishing between encoder and decoder model families (Section 2.1.3). Finally, we examine the scaling of these architectures into LLMs and the innovations that make their local deployment feasible (Section 2.1.4).

2.1.1 The Shift to Learning-Based Language Processing

NLP has historically relied on *hand-crafted rules* and *statistical models*, such as rule-based parsers, n-gram language models, and linear classifiers with manually engineered features, trained from scratch on labeled data for each individual task [Jurafsky and Martin, 2026]. Both approaches can require substantial domain expertise and large task-specific datasets, resources that may be unavailable for specialized applications such as occupation coding. The introduction of PLMs fundamentally changed this paradigm by shifting from task-specific feature engineering to generic representation learning [Min et al., 2023].

The core idea of *pre-training* is to train a single large model on a massive corpus of unlabeled text, enabling it to acquire a general-purpose representation of language,

and then adapt or fine-tune that representation to downstream tasks with comparatively little labeled data [Min et al., 2023; Radford and Narasimhan, 2018]. The architecture that made this strategy both practical and extraordinarily effective is the *transformer* [Vaswani et al., 2017]. Before transformers, *recurrent neural networks* processed text sequentially, one token at a time, which limited parallelization because each step depended on the previous hidden state and made it difficult to learn long-range dependencies due to vanishing or exploding gradients [Bengio et al., 1994; Hochreiter and Schmidhuber, 1997]. Transformers resolved both problems simultaneously by replacing sequential recurrence with a mechanism that lets every token in the input directly interact with every other token, an operation known as *self-attention* [Vaswani et al., 2017]. This design is more expressive for capturing long-distance linguistic structure and far more parallelizable on modern hardware, enabling the scale of computation that defines contemporary AI [Brown et al., 2020; Vaswani et al., 2017].

2.1.2 The Transformer Architecture

To understand how a transformer processes language, we trace the data step by step from raw text to a final token prediction. In this section, we outline this pipeline, detailing the initial conversion of text into numerical embeddings (Section 2.1.2.1), the self-attention mechanism that contextualizes these representations (Section 2.1.2.2), the layers that inject factual knowledge (Section 2.1.2.3), and how these components are stacked to generate sequential output (Section 2.1.2.4).

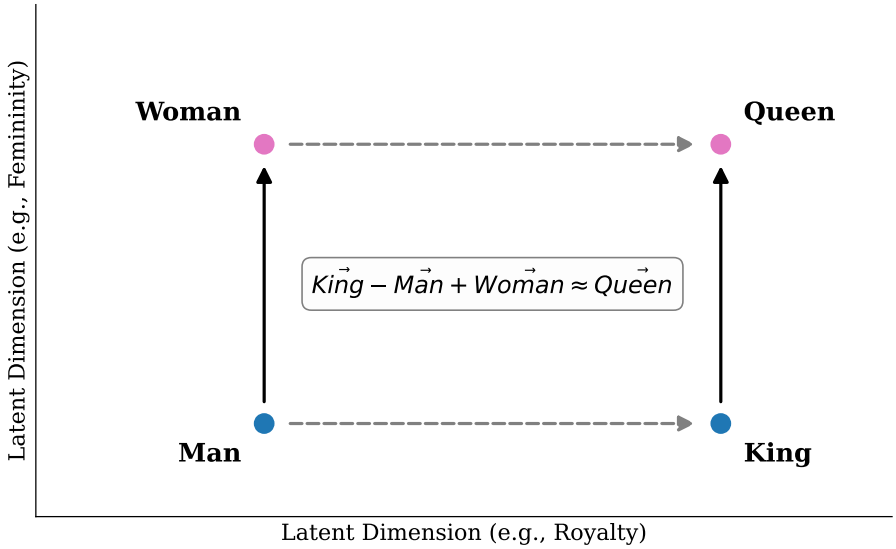
2.1.2.1 Tokenization and Word Embeddings

The first step converts raw input text into a sequence of discrete units called *tokens*. Tokens are typically words, sub-words, or common character sequences; transformer models often use vocabularies with tens of thousands of entries, such as the 50,257-token vocabulary of GPT-3 [Brown et al., 2020; Sanderson, 2024]. Each token is then mapped to a high-dimensional numerical vector, its *embedding*, via a learned embedding matrix. This matrix assigns a unique vector to every vocabulary item and is updated throughout training [Sanderson, 2024; Vaswani et al., 2017].

A key insight is that the geometry of the resulting embedding space is not arbitrary. As the model learns to predict text, it adjusts the embedding matrix so that directions in the space carry semantic meaning. A canonical example is the gender direction: the vector difference between the embeddings of woman and man closely parallels that between queen and king, allowing analogies such as $\vec{\text{king}} - \vec{\text{man}} + \vec{\text{woman}} \approx \vec{\text{queen}}$ [Mikolov et al., 2013b; Sanderson, 2024], as illustrated in Figure 2.1. In the occupational domain, embeddings of job titles whose professions share similar skill profiles cluster in nearby regions of this space, providing the geometric foundation for the semantic retrieval described in Section 2.3.

However, these initial embeddings encode only the meaning of an isolated token. The same word, say "Leiter", receives an identical initial embedding whether it refers to a manager or a ladder, because this representation does not yet take the surrounding sentence into account. All context-sensitivity is introduced by the subsequent transformer layers [Sanderson, 2024; Vaswani et al., 2017].

Figure 2.1: Geometric Relationships in Semantic Word-Embedding Space. A conceptual 2D projection based on the canonical example by Mikolov et al. [2013b]. The spatial arrangement demonstrates how semantic relationships—such as gender or royalty—are encoded as offsets, allowing word analogies to be solved via simple vector algebra ($\vec{\text{King}} - \vec{\text{Man}} + \vec{\text{Woman}} \approx \vec{\text{Queen}}$).



2.1.2.2 Self-Attention: How Words Inform Each Other

The central mechanism of the transformer is self-attention, depicted as the "Multi-Head Attention" blocks in Figure 2.2. Its purpose is to update each token's initial, context-independent embedding by incorporating relevant information from the surrounding tokens in the sequence [Sanderson, 2024; Vaswani et al., 2017].

For each token, the model produces three vectors by multiplying its current embedding with three learned weight matrices: a *query* (what information this token is seeking from context), a *key* (what information this token makes available to others), and a *value* (the information it would contribute if selected). The model then computes the dot product between each token's query vector and every other token's key vector. A high dot product indicates that two tokens are mutually relevant. Applying a softmax function normalizes these scores into attention weights—a probability distribution over the full token sequence. Finally, the token's embedding is updated as a weighted sum of the value vectors, with the attention weights as coefficients [Sanderson, 2024; Vaswani et al., 2017].

The result is contextually enriched representations. Whereas in Section 2.1.2.1 we assigned a single static vector to each word type, self-attention produces a distinct contextualized vector for each occurrence. A token for the word "head", for instance, attends strongly to "department" or "leadership" in a managerial context, shifting its embedding toward a leadership region of semantic space; in an anatomical context it attends to "shoulders" or "neck", yielding an entirely different representation. This mechanism resolves the terminological ambiguity that makes survey-based occupation coding challenging (Section 1.1.2).

Empirical analyses have shown that different attention heads specialize in distinct relational patterns or abstract concepts, including syntactic dependencies, coreference, and other semantic relations [Clark et al., 2019; Tenney et al., 2019]. In practice, a transformer runs multiple independent attention heads in parallel, each trained to capture a different type of abstract concept. Their outputs are projected and combined to produce the updated embedding passed to the next layer [Sanderson, 2024; Vaswani et al., 2017].

2.1.2.3 Multi-Layer Perceptrons: Where Knowledge Lives

Following each attention block, every token embedding passes independently through a multi-layer perceptron (MLP), visualized as the "Feed Forward" sub-layer in Figure 2.2. Unlike the attention mechanism, the MLP does not allow tokens to communicate with one another; each vector is processed by the same function in parallel [Sanderson, 2024; Vaswani et al., 2017].

The MLP consists of two learned matrix multiplications with a non-linear activation function between them. The first matrix expands the embedding into a higher-dimensional intermediate space; the activation function, commonly rectified linear unit (ReLU) [Glorot et al., 2011] or its smoother variant gaussian error linear unit (GELU) [Hendrycks and Gimpel, 2016], introduces non-linearity, which enables the network to approximate complex, non-linear functions beyond what a purely linear transformation could represent [Cybenko, 1989; Hornik et al., 1989]. The second matrix projects the result back to the original embedding dimension [Sanderson, 2024; Vaswani et al., 2017].

Intuitively, the neurons in the intermediate space act as *soft pattern detectors* [Sanderson, 2024]. For example, a neuron whose input weights align with the combination of the context terms "software" and "development" might activate strongly when the embedding encodes both; the corresponding output weights can then add a direction into the space that encodes the associated occupation "software engineer". A growing body of work formalizes this intuition, showing that transformer feed-forward layers effectively implement key-value memories, where specific input patterns trigger the retrieval of stored output vectors [Geva et al., 2021]. Empirically, MLP blocks account for roughly two-thirds of all transformer parameters in large models such as GPT-3, where they constitute on the order of 10^{11} parameters [Brown et al., 2020; Sanderson, 2024]. Together, these findings support the view that MLP blocks serve as a primary repository of factual world knowledge in contemporary LLMs.

2.1.2.4 Stacking Layers and the Flow of Meaning

A complete transformer interleaves multiple paired attention and MLP blocks sequentially, denoted by the "$N\times$" repetitions in Figure 2.2. After each block, the token embeddings become progressively richer: the final-layer representation of a token encodes not just its isolated meaning but its meaning as shaped by the entire surrounding passage. The end-to-end process can be summarized in three steps: (1) tokenize the input and look up each token's initial embedding; (2) pass the sequence through alternating attention blocks (context integration) and MLP blocks (factual enrichment), repeated across many layers; and (3) at the final layer, project the

representation of the prediction position through a learned un-embedding matrix to obtain a probability distribution over the vocabulary, from which the most likely next token is sampled. Generating a complete response repeats this loop token by token, appending each new token to the growing context until a stopping condition is met [Sanderson, 2024; Vaswani et al., 2017].

2.1.3 Encoder and Decoder Architectures

Transformer-based models divide into two families based on their pre-training objective and attention pattern. *Encoder models* such as BERT are trained on *masked language modeling* [Devlin et al., 2019]: some tokens are randomly replaced with a mask token, and the model learns to reconstruct them using both left and right context simultaneously. Because every token can attend to its full surrounding sequence bidirectionally, encoder models produce deeply contextualized representations well-suited for tasks requiring comprehension of the complete input, including classification, semantic similarity, and named-entity recognition [Devlin et al., 2019; Reimers and Gurevych, 2019; Taillé et al., 2020]. The hidden states of an encoder serve as the primary raw material for text embedding models (Section 2.2).

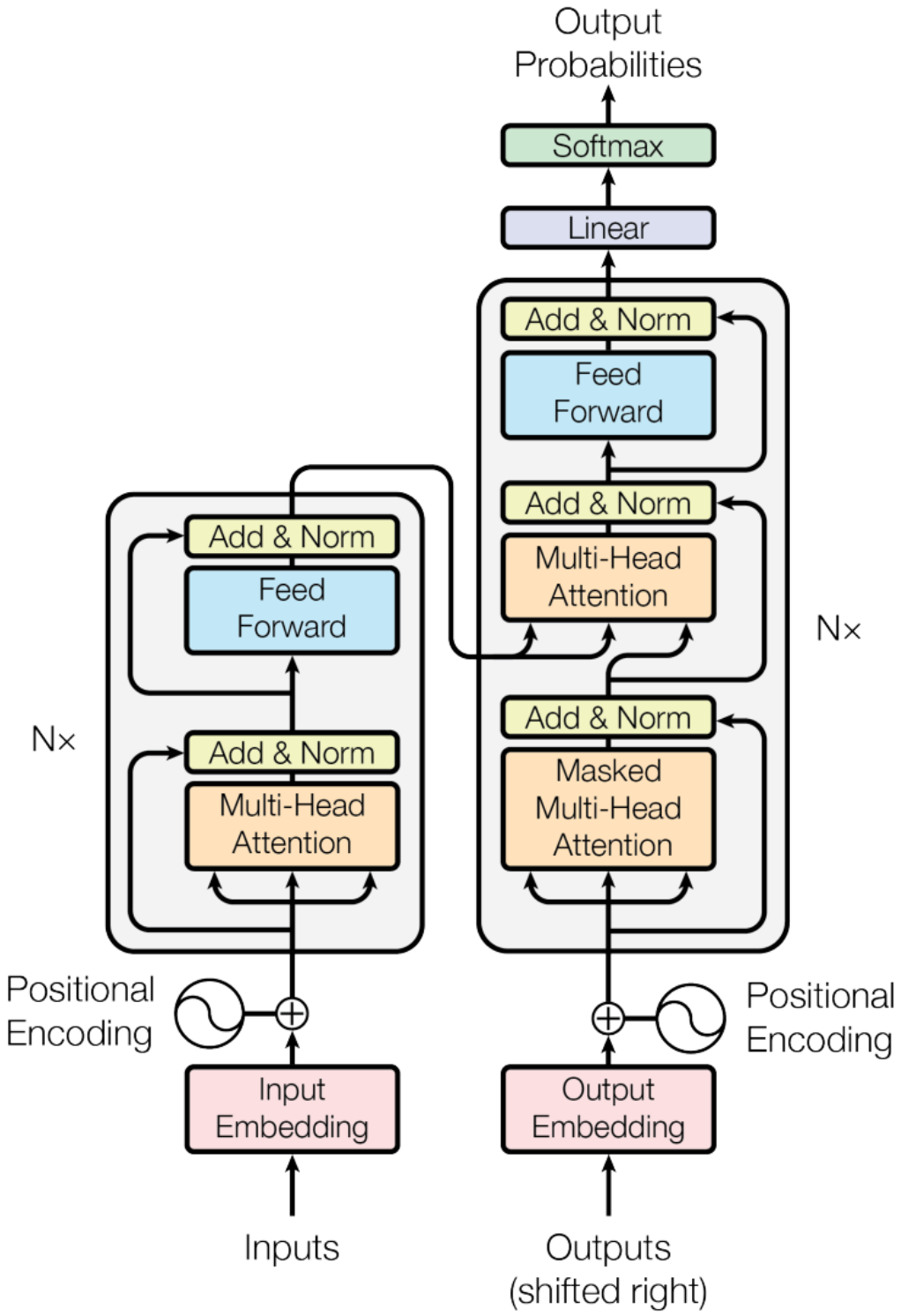
Decoder models such as GPT use a *causal attention mask* [Radford and Narasimhan, 2018]: each token can only attend to preceding tokens, preserving the left-to-right generation order required for autoregressive text production [Vaswani et al., 2017]. Pre-training on next-token prediction over vast text corpora equips decoder models with powerful generative capabilities [Radford and Narasimhan, 2018]. Modern LLMs—including Qwen3, Magistral, Phi-4, and DeepSeek-R1—are decoder models scaled to billions of parameters and instruction-fine-tuned to follow natural-language task specifications [Abdin et al., 2024; Guo et al., 2025; Min et al., 2023; Mistral-AI et al., 2025; Yang et al., 2025].

2.1.4 Large Language Models at Scale

An LLM is a transformer trained at extreme scale: billions of parameters on trillions of training tokens. Scaling produces qualitative changes in capability [Wei et al., 2022], including *in-context learning* [Brown et al., 2020]: the ability to perform a task by conditioning on instructions or demonstrations provided within the prompt, without any weight update. This encompasses zero-shot settings, where only a task instruction is given, as well as one- and few-shot settings, where a small number of input–output examples are additionally included in the prompt. The query-rewriting and classification stages of our approaches exploit both forms, providing the LLM with structured German-language instructions—with and without in-prompt examples—to rewrite survey data and assign KldB codes, without any task-specific fine-tuning.

Two architectural and quantization innovations extend the deployment feasibility of LLMs under hardware constraints. MoE architectures [Yang et al., 2025; Zhang et al., 2025a] route each token through only a learned subset of feed-forward experts per layer, enabling very large total parameter counts at reduced per-token computation. *Quantization* compresses model weights from 32- or 16-bit to 4-bit floating-point representations, reducing memory requirements by over 50% at modest quality cost [Husom et al., 2025]. Together, these advances allow 14B–32B-parameter models to run on consumer-grade hardware—a prerequisite for the data-privacy-constrained local deployment in this thesis (Section 4.1.3).

Figure 2.2: Original Transformer Model Architecture. The foundational encoder-decoder structure introduced by Vaswani et al. [2017]. The left block depicts the encoder, which processes the input sequence, while the right block depicts the autoregressive decoder. Models like BERT utilize only the encoder stack, whereas models like GPT rely entirely on the decoder. Image sourced from Mao [2019], available at: https://leimao.github.io/blog/Transformer-Explained/.



2.2 Text Embedding Models

While LLMs generate text token by token, embedding models (EMs) convert text into fixed-length numerical vectors that capture semantic content in a form amenable to mathematical comparison. We begin by tracing the evolution from static to contextual text representations (Section 2.2.1), then explain how the geometry of embedding spaces enables similarity measurement via cosine similarity (Section 2.2.2). We subsequently introduce the dual-encoder architecture that makes vector-based retrieval scalable and practical (Section 2.2.3), before discussing multilingual and instruction-finetuned variants relevant to the retrieval component of our RAG systems (Section 2.2.4).

2.2.1 From Static to Contextual Text Representations

Early word-embedding methods such as Word2Vec assigned a single static vector to each word, independent of context [Mikolov et al., 2013a]. Transformer-based EMs like BERT resolved this limitation: because self-attention is context-sensitive, tokens representing the same word receive different vector representations depending on their surrounding context. A *sentence encoder* distills the sequence of token vectors produced by the transformer into a single fixed-length vector representing the entire passage. A common strategy is *mean pooling*, which averages the final-layer token embeddings [Reimers and Gurevych, 2019]. Specialized training objectives, particularly *contrastive learning* that pulls embeddings of semantically similar texts closer while pushing dissimilar texts apart [Gao et al., 2021], further sharpen these representations for similarity-based retrieval [Karpukhin et al., 2020].

2.2.2 Semantic Geometry and Cosine Similarity

Sentence-embedding spaces inherit the geometric properties of their underlying transformer representations: semantically related texts cluster near one another, and the standard metric for closeness is *cosine similarity*—the cosine of the angle between two embedding vectors [Reimers and Gurevych, 2019]. A similarity of 1 indicates parallel vectors (near-perfect semantic overlap); a similarity of 0 indicates orthogonal vectors (unrelated content).

Cosine comparison is computationally inexpensive. Once all documents in a knowledge base have been embedded and stored, retrieving the most relevant match for a new query reduces to finding the nearest neighbors of the query vector—a task that modern approximate nearest-neighbor libraries perform in milliseconds even over millions of documents [Bruch, 2024; Douze et al., 2024; Johnson et al., 2021]. For occupation coding, this means that an informal survey response can be semantically matched to a KldB description it does not lexically resemble, provided both map to nearby regions of the embedding space.

Despite these advantages, cosine similarity has well-documented limitations rooted in the distributional nature of embedding training. Sentence vectors are dominated by topical and lexical signals, causing semantically local operators such as negation to have negligible effect on the global representation—a phenomenon demonstrated by Vahtola et al. [2022], who show that contextual embeddings treat “I am guilty” and “I

am not guilty" as near-paraphrases. More fundamentally, high cosine similarity scores do not guarantee classification correctness: Steck et al. [2024] show analytically that cosine similarity can yield arbitrary results depending on training-objective regularization, and Chen et al. [2025] demonstrate empirically through their "Information Loss Funnel" that vector proximity does not align with task-specific utility—a system achieving near-perfect synthetic retrieval recall can simultaneously achieve near-zero label recall when the similarity metric fails to reflect task-relevant distinctions. For occupation coding, this means that the top-ranked retrieved KldB candidate—the document with the highest cosine score—is not reliably the correct classification target: occupational descriptions within the same subgroup share nearly identical task vocabulary, placing them in proximal embedding regions with near-identical scores that cosine comparison alone cannot discriminate. This limitation motivates the LLM-based classification stage (Section 4.3.2), which applies semantic reasoning across all top-k retrieved candidates rather than directly returning the highest-scoring document.

2.2.3 Dual-Encoder Architecture for Efficient Retrieval

The *dual-encoder* architecture [Karpukhin et al., 2020; Reimers and Gurevych, 2019] is the standard design for retrieval-oriented EMs. Two encoder networks—one for queries and one for documents—process their respective inputs independently and produce embeddings in a shared vector space. Relevance is computed as the dot product or cosine similarity between the resulting vectors.

The critical advantage is *pre-computability*: because the two encoders operate independently, all document embeddings can be computed once offline and stored in a vector index. At retrieval time, only the query needs to be encoded on the fly; the pre-computed index is then searched using an approximate nearest-neighbor structure such as hierarchical navigable small world (HNSW) [Bruch, 2024; Malkov and Yashunin, 2020]. This separation of offline indexing from online retrieval makes vector-based RAG both scalable and practical.

2.2.4 Multilingual and Instruction-Finetuned Embeddings

Multilingual EMs such as Multilingual-E5 [Wang et al., 2024b] and Jina-Embeddings-v3 [Sturua et al., 2024] are trained on diverse cross-lingual corpora, enabling robust semantic matching across different languages and registers. *Instruction-finetuned EMs* [Su et al., 2023] extend this capability by accepting a natural-language prefix prepended to the input that steers the representation toward task-relevant features, for example specifying "Represent this German job description for retrieval against occupation taxonomy entries" to focus the embedding on occupational semantics rather than general linguistic form. Most recently, EMs built directly on LLM foundations [Wang et al., 2024a], such as Qwen3-Embedding [Zhang et al., 2025b], inherit the deep multilingual understanding and factual knowledge of their underlying decoder model while retaining the inference efficiency of a dual-encoder, achieving state-of-the-art performance on multilingual retrieval benchmarks.

2.3 Retrieval-Augmented Generation

RAG [Lewis et al., 2020] combines the generative capabilities of LLMs with an external, non-parametric knowledge base to produce more factual, grounded, and updatable outputs. We begin by motivating why this architecture is necessary by examining the limits of purely parametric knowledge (Section 2.3.1). We then describe the core pipeline components that constitute a RAG system (Section 2.3.2). Finally, we discuss how LLM-based query rewriting can improve retrieval by adapting user queries to the structure of the knowledge base (Section 2.3.3).

2.3.1 Motivation: Limits of Purely Parametric Knowledge

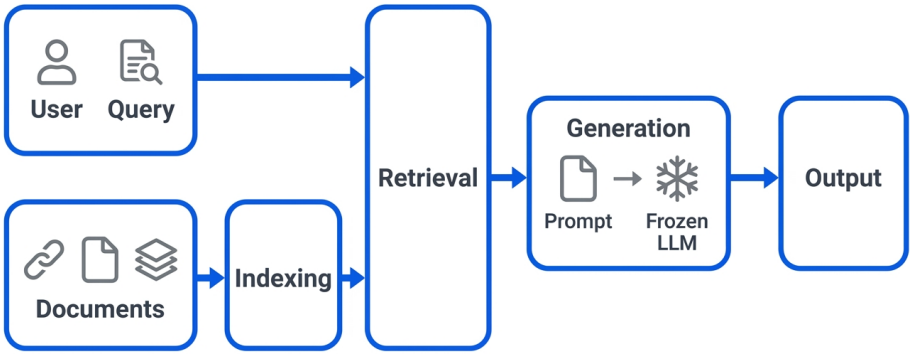
LLMs encode the knowledge seen during pre-training directly in their parameter weights—a form of storage called *parametric memory* [Lewis et al., 2020]. This approach has two fundamental limitations in the context of occupation coding. First, parametric knowledge is fixed at training time and cannot be updated without retraining; a taxonomy revision or the emergence of new occupations would require the model to be retrained from scratch [Lewis et al., 2020]. Second, LLMs—especially smaller models—often lack reliable knowledge of specialized, domain-specific taxonomies. Achananuparp et al. [2025] demonstrate this concretely: on tasks requiring occupation-code generation without external hints, the Llama3-8B model achieves only 0.11 accuracy on the O*NET–SOC taxonomy, averaging only 0.28 accuracy across all knowledge-assessment tasks—yet the same model reaches 0.76 accuracy when supported by retrieval from the same taxonomy within a RAG pipeline.

RAG addresses both limitations by separating knowledge from reasoning [Lewis et al., 2020]. A dedicated external knowledge base stores the authoritative taxonomy documents; the LLM is responsible only for understanding and generation, not for memorizing taxonomy specifics. This separation enables straightforward knowledge-base updates without any model retraining, a key advantage in domains where taxonomies evolve over time. Additionally, because retrieved documents are visible in the prompt, the model's outputs become more interpretable and less prone to hallucination [Shuster et al., 2021].

2.3.2 The RAG Pipeline

At a high level, a RAG system consists of three interacting components: a knowledge base and its associated vector index, a retriever, and a generator. The *knowledge base* is a collection of documents that encode the domain knowledge required for the task. Prior to deployment, each document is embedded using a pre-trained EM and stored in a *vector index*, so that semantic similarity search can operate over fixed-length representations rather than raw text—a process corresponding to the indexing stage shown in Figure 2.3. Given a query at inference time, the *retriever* encodes the query into the same embedding space as the document vectors and identifies the k most similar items using approximate nearest-neighbor search—the retrieval stage of Figure 2.3. These top-k candidates are retrieved and assembled, together with the original query and a task-specific instruction, into a structured prompt. The *generator* is an LLM whose weights remain frozen throughout inference; drawing on both the retrieved context and its parametric knowledge, it produces an answer grounded in the external knowledge base, such as an assigned occupation code or a natural-language justification [Gao et al., 2024; Lewis et al., 2020].

Figure 2.3: Conceptual Framework of a RAG Pipeline. The figure depicts the three stages of a RAG system: (1) *Indexing*, in which source documents are embedded and stored in a vector index offline; (2) *Retrieval*, in which an incoming user query is matched against the index to obtain the most relevant documents; and (3) *Generation*, in which the retrieved documents and the original query are assembled into a prompt and passed to a frozen LLM—frozen indicating that the model's weights are not updated at inference time. Adapted from Gao et al. [2024]; figure redesigned using Perplexity AI (Google Gemini image generation).



2.3.3 LLM-Based Query Rewriting

Even with strong embedding models, the register mismatch between informal user inputs and formal knowledge-base documents can hinder retrieval. The original query may be terse, ambiguous, or phrased in a way that does not resemble any document in the vector index. Query rewriting addresses this mismatch by transforming a query before retrieval to improve its alignment with the target documents [Carpineto and Romano, 2012]. Applied to RAG, LLM-based query rewriting leverages the generative capabilities of language models to reformulate queries in ways that are better suited to the knowledge base [Koo et al., 2024; Ma et al., 2023].

A particularly effective strategy is *hypothetical document generation* [Gao et al., 2023]: instead of embedding the original query directly, the LLM generates a hypothetical document that plausibly answers the query, capturing the stylistic and semantic patterns of real knowledge-base documents even if it contains unverified details. This hypothetical document is then encoded into an embedding vector, which is used in place of the original query to retrieve the most similar real documents from the index. Because the generated text is stylistically closer to documents in the knowledge base than the original user query, its embedding aligns more tightly with the target document's embedding, substantially improving retrieval precision [Gao et al., 2023]. For occupation coding, generating hypothetical job tasks in the formal style of KldB descriptions, rather than embedding an informal survey response directly, is expected to reduce both register and granularity mismatches (Section 1.1.2) between the query representation and the underlying taxonomy.

2.4 Summary

This chapter introduces the three technical building blocks on which the remainder of the thesis is constructed: pre-trained transformer-based language models, text embedding models, and retrieval-augmented generation.

The first block traces the evolution from classical, rule-based NLP toward large-scale pre-training on unlabeled text. The chapter explains how the transformer architecture uses self-attention to capture contextual relationships across an entire input sequence, and how stacking attention and feed-forward layers enables a model to build progressively abstract linguistic representations. It then distinguishes between encoder-only architectures, which are optimized for contextual text understanding, and decoder-based LLMs, which perform autoregressive generation and exhibit emergent in-context learning at sufficient scale. Recent advances in mixture-of-experts sparsity and low-bit quantization are highlighted as the factors that bring large-parameter models within reach of local, consumer-grade hardware deployments.

The second block addresses text embedding models, which project variable-length text into dense numeric vectors such that semantically similar passages cluster together in geometric space. The chapter contrasts static word-level embeddings with contextual representations, motivates cosine similarity as a practical semantic distance metric, and presents the dual-encoder design—separate query and document encoders—as the architecture that makes approximate nearest-neighbor search tractable at index scale. Multilingual and instruction-tuned embedding models receive particular attention, as they enable robust semantic matching across different languages, and accept task-specific instruction prefixes that steer the representation toward a desired retrieval objective.

The third block introduces retrieval-augmented generation as the paradigm that decouples factual knowledge from a model's parametric memory. The chapter describes the core RAG pipeline—offline document indexing, online dense retrieval, and conditioned generation over retrieved context—and motivates this separation on two grounds: the knowledge base can be updated without retraining the model, and grounding generation in retrieved documents improves interpretability while reducing hallucination. The chapter closes with LLM-based query rewriting, covering hypothetical document generation as the key strategy: an LLM generates a synthetic document that resembles what a relevant knowledge-base entry might look like, and that synthetic text embeds more tightly with actual indexed documents than the original, often underspecified query does—a strategy this thesis adapts as domain-specific hypothetical job-task generation in the QR-RAG pipeline.

Together, these foundations equip the reader with the conceptual vocabulary required to follow the system design, experimental decisions, and result interpretation in the chapters that follow.

3. DZHW Survey Data: Quality Assessment, Preprocessing, Dataset Construction

Survey-based occupation coding relies on naturalistic free-text responses that exhibit systematic quality challenges—including misplaced task descriptions, unknown occupational abbreviations, and multi-role entries. Evaluating RAG systems on such data requires distinguishing preprocessing-addressable syntactic problems from semantic challenges that systems must handle during inference, yet these distinctions are often made ad hoc rather than through systematic frameworks. Without principled classification of data-quality issues, preprocessing risks either introducing evaluation bias through excessive data cleaning or distorting performance metrics through insufficient ground-truth correction.

This chapter develops and applies a data-quality assessment framework (DQAF) that addresses this gap through systematic issue classification along three dimensions—issue origin, rule-based preprocessing feasibility, and LLM-addressability. This framework enables principled decisions about which issues to resolve through deterministic preprocessing and which to preserve for authentic RAG-system evaluation, ensuring that performance metrics reflect genuine semantic competence rather than artificially simplified scenarios.

We first characterize the raw DZHW survey dataset and identify surface-level quality issues (Section 3.1), then develop the data-quality assessment framework (DQAF) with its three dimensions (Section 3.2). Applying this framework, we classify 13 distinct issues into three treatment classes: five preprocessing-addressable issues, six LLM-addressable issues intentionally preserved, and two inherent dataset characteristics (Section 3.3). We implement a 6-step preprocessing pipeline that systematically resolves addressable issues while establishing consistent ground-truth conditions (Section 3.4), then analyze the resulting preprocessed dataset to identify structural challenges that inform our sampling strategy (Section 3.5). Finally, we construct four evaluation datasets using stratified sampling: one maximizing taxonomic breadth

across 1,088 occupational codes, and three inducing controlled input-complexity conditions, validated through quantitative and qualitative assessment (Section 3.6).

This systematic approach yields evaluation datasets that balance computational feasibility with taxonomic representativeness and authentic linguistic complexity, establishing a rigorous foundation for RAG-system assessment. By preserving LLM-addressable challenges while removing syntactic irregularities, these datasets enable meaningful evaluation of system robustness under realistic survey-data conditions.

3.1 Raw Survey Dataset

This section characterizes the raw DZHW survey dataset across three dimensions: its provenance and survey context (Section 3.1.1), its structure and field composition (Section 3.1.2), and prevalent surface-level data-quality issues that motivate systematic preprocessing (Section 3.1.3).

3.1.1 Origins and Survey Context

Safikhani et al. [2023] leveraged DZHW survey data for occupation-coding research, underscoring the necessity of careful data preparation. We obtained a comparable dataset from DZHW based on their work. The data originate from two longitudinal survey series [Fabian et al., 2023; Hoffstätter et al., 2021]: the *Graduate Survey Series* (initiated in 1989) and the *Survey Series of School Leavers with Higher Education Entrance Qualifications* (initiated in 1976).

The Graduate Survey Series tracks graduates at three intervals (1–1.5 years, 5 years, and 10 years post-graduation), examining career trajectories, assessing skill mismatches between higher education and employment, and monitoring occupational transitions. The School Leavers series follows educational and professional pathways of those completing higher education entrance qualifications. Both series collect occupational data through German free-text responses to open-ended questions, capturing naturalistic expressions of job roles as individuals perceive and articulate them.

3.1.2 Structure and Field Composition

We received two anonymized comma-separated values (CSV) files from DZHW and focused exclusively on the training set, which contains 52,542 instances across nine columns—sufficient data for robust evaluation. Each instance contains four field types (detailed in Table 3.1): non-standardized textual inputs comprising (i) job-title descriptions and (ii) two job-task descriptions; (iii) KldB codes assigned by trained coders using the KldB 2010 taxonomy; and (iv) synthetic fields generated by Safikhani et al. [2023] for supervised occupation coding. Upon inspection, the task fields revealed diverse content including work activities, responsibilities, qualifications, hierarchical positions, and industry context. We excluded the synthetic fields from subsequent analysis and preprocessing due to their artificial origin, opaque processing methods, and redundancy with original content fields.

Table 3.1: Raw DZHW Survey Dataset Structure and Representative Examples. Surface-level quality issues include character-encoding errors transforming "Geschäftsführer" into "Gesch?ftsf?hrer"; systematic placeholders ("-7", "???") replacing substantive content; code-format deviations ("1402" instead of "01402"). Synthetic fields shown for completeness but excluded from subsequent analysis and preprocessing.

Original Name	Readable Name	Examples
Content fields:		
cberuf	Job Title	"POLIZIST", "AERZTIN", "Gesch?ftsf?hrer", "TM1 Berater", ...
cfunkti1	Tasks 1	"NAEHERIN", "QM", "siehe oben", "-1 keine Angabe", ...
cfunkti2	Tasks 2	"Rechtsprechung", "Personalführung", "-7", "-9", "???", "k.A.", ...
kldb	KldB Code	84304, 1402, 99999, -1, "-1. keine Angabe", "", ...
DZHW-generated synthetic fields (excluded from analysis and preprocessing):		
—	Synthetic 1	22, 27, 41, ...
—	Synthetic 2	9810, 20711, 34902, ...
—	Synthetic 3	"FACHARZT CHIRURGIE", "VORSCHULLEHRERIN", ...
—	Synthetic 4	"ZIMMERMANN", "REFERNTIN #LEERZEICHEN# ...", ...
—	Synthetic 5	"WISS MIT", "SENIOR CONTROLLER", ...

3.1.3 Surface-Level Data-Quality Issues

Having established the dataset's structure and field composition, we examined the quality characteristics that necessitate systematic preprocessing. Safikhani et al. [2023] observed that participants often provide brief, incomplete descriptions lacking rich linguistic features. Table 3.1 presents the dataset structure with original and readable column names alongside representative examples. These examples reveal both typical German survey responses and prevalent surface-level data-quality issues such as character-encoding errors, systematic placeholders, and inconsistent KldB-code formats.

These data-quality issues span multiple fields across all 52,542 instances, making manual correction infeasible. We therefore address them systematically through three steps that structure our subsequent analysis. First, we construct the DQAF as a structured classification framework that distinguishes preprocessing-addressable syntactic issues from deeper semantic challenges (Section 3.2). Second, we apply this framework to classify all 13 identified issues, determining which can be resolved through rule-based methods and which must be preserved for RAG-system evaluation (Section 3.3). Third, we develop and apply a rule-based preprocessing pipeline constrained by our computational resources (Appendix A.2.1) that systematically resolves addressable issues (Section 3.4), ultimately enabling the construction of four evaluation datasets tailored to different analytical objectives (Section 3.6).

3.2 Data-Quality Assessment Framework

Having characterized the raw dataset and identified prevalent quality issues, we developed a systematic framework for classifying these issues to determine appropriate

treatment strategies. Manual correction of individual instances is infeasible at scale for this thesis. Like Safikhani et al. [2023], who applied preprocessing to similar DZHW data, we recognized that systematic data preparation is essential prior to RAG evaluation—but this preparation must preserve evaluation authenticity. We therefore conducted preliminary manual analysis of recurring issue patterns across the dataset, which revealed 13 distinct issues from which we developed an original DQAF.

The framework accomplishes two objectives: (i) it separates rule-based preprocessing from advanced semantic-processing capabilities reserved for our RAG systems, and (ii) it ensures that performance metrics reflect genuine semantic competence under authentic data conditions rather than artificially simplified scenarios induced by excessive preprocessing. The framework organizes issues along three dimensions—*issue origin* (Section 3.2.1), *rule-based preprocessing feasibility* (Section 3.2.2), and *LLM-addressability* (Section 3.2.3)—with Section 3.3 applying the framework to classify the 13 identified issues.

3.2.1 Issue-Origin Dimension

The issue-origin dimension classifies data-quality problems according to their source, tracing them from initial survey responses through manual occupation coding. We define three categories: (i) *Survey-response patterns*, which stem from authentic participant behavior during survey completion and reflect genuine interactions with data-collection instruments; (ii) *Data-processing artifacts*, which arise during technical handling—such as data transfer, storage, or format conversion—and introduce systematic errors; and (iii) *Manual-coding procedures*, which originate in human-annotation activities and data-collection protocols, reflecting inherent variability and complexity in survey-based occupation coding. Together, these categories establish a provenance layer that informs subsequent classification decisions and supports traceability of issue origins.

3.2.2 Rule-Based Preprocessing Feasibility Dimension

The rule-based preprocessing feasibility dimension evaluates whether a given issue can be resolved through deterministic techniques within typical computational constraints. Rule-based preprocessing employs methods such as pattern matching, format standardization, robust parsing, systematic filtering, and cross-reference validation—approaches that are deterministic, reproducible, and do not require advanced semantic understanding, extensive domain expertise, or substantial computational resources. This distinguishes them from resource-intensive semantic-processing approaches, including the use of LLMs.

We define two categories: (i) *Preprocessing-addressable issues* are those that can be systematically corrected using these rule-based techniques once the problematic patterns (e.g., recurring placeholder values such as "-7" or "???") are identified; and (ii) *Preserved issues*, which exceed the scope of rule-based correction and thus require further classification via the LLM-addressability dimension (Section 3.2.3). This distinction enables a clear separation between issues suitable for systematic, reproducible correction and those that must be retained as challenges for semantic handling during RAG-system operation.

3.2.3 LLM-Addressability Dimension

The LLM-addressability dimension distinguishes preserved issues based on their relationship to advanced semantic-processing capabilities. Analysis of preserved issues that exceed rule-based correction revealed a critical distinction between those that systems can potentially handle and those that represent irreducible evaluation constraints.

We define two categories: (i) *LLM-addressable issues* are those that require contextual interpretation, semantic normalization, or domain-specific knowledge—capabilities that can be supported in principle by LLMs during system operation. Examples include respondents' expression variability and unknown occupational abbreviations. These issues were intentionally preserved in the preprocessed dataset to evaluate system robustness under authentic conditions. (ii) *Inherent dataset characteristics* represent fundamental limitations in data collection or human annotation—such as multi-role entries where participants describe multiple occupations per response, or multi-label classification patterns where expert coders assign inconsistent codes to identical descriptions—that currently no known automated approach, including LLMs, can systematically resolve. These require expert authority and access to original coding rationale, representing boundaries for evaluation rather than targets for system improvement.

To preserve evaluation authenticity, our preprocessing remained strictly rule-based (as defined in Section 3.2.2), while LLM-based semantic processing was reserved exclusively for system-level functions (e.g., query rewriting in QR-RAG, Section 1.3.2). This ensures that performance metrics reflect true semantic competence rather than artificially simplified scenarios induced by excessive preprocessing. Together, these categories complete the framework by distinguishing legitimate targets for semantic handling from unavoidable constraints that define the boundaries of valid evaluation.

3.3 Data-Quality Issues and Classification

Having established the DQAF with its three dimensions, we applied this framework to classify the 13 data-quality issues identified in preliminary analysis. Issues are organized by final treatment class—preprocessing-addressable (Section 3.3.1), LLM-addressable preserved (Section 3.3.2), and inherent dataset characteristics (Section 3.3.3)—and summarized in Section 3.3.4.

3.3.1 Preprocessing-Addressable Issues

The following five issues are characterized by their syntactic nature, enabling deterministic correction through rule-based preprocessing within our computational constraints. We addressed them systematically in our preprocessing pipeline (Section 3.4) to improve data quality and remove noise that would otherwise distort evaluation metrics.

3.3.1.1 Non-Informative Participant Responses

We identified non-informative participant responses originating from survey-response patterns in which referential placeholders—such as "siehe oben" (see above) or "keine

Angabe" (no information)—replaced substantive occupational content. These entries lack meaningful occupational information, may degrade data quality, and introduce noise that can impair system performance. Because this issue is syntactic and fully amenable to rule-based correction, we classified it as preprocessing-addressable. Treatment involves cataloging such placeholders through manual inspection and removing them via pattern matching and systematic filtering, which can enhance data quality.

3.3.1.2 Structural Data Corruption

We identified structural data corruption stemming from data-processing artifacts in which malformed CSV entries disrupted parsing. For example, a line containing ""Eventmanagement";" Krisenmanagement"" triggered parsing errors in `pandas.read_csv()`[10], producing messages like "Expected 1 fields in line 112, saw 7". These misplaced delimiters and inconsistent quoting break field boundaries, corrupt the dataset structure, and prevent downstream analysis. Because this issue is purely syntactic and arises from predictable formatting errors, we classified it as preprocessing-addressable. Treatment options include robust parsing techniques—such as using error-handling modes or custom delimiters—that accommodate malformed entries without content loss, restoring structural integrity and enabling reliable processing and evaluation.

3.3.1.3 Inconsistent Code Formats

We identified inconsistent code formats resulting from manual-coding procedures in which occupation codes deviated from the standard 5-digit KldB format. Examples included 4-digit military codes like "1402" (missing the leading zero, thus requiring "01402") or invalid code entries such as "-1". These code-format problems create non-standard targets and would mislead evaluation by marking correct predictions as incorrect, thereby compromising the reliability of performance assessment. Because this issue involves syntactic deviations that are fully predictable, we classified it as preprocessing-addressable. Treatment involves standardizing codes by adding leading zeros where necessary and removing entries with invalid formats, which ensures that respective ground-truth labels conform to the official taxonomy, improving consistency and fairness in evaluation.

3.3.1.4 Invalid Taxonomy Codes

We identified invalid taxonomy codes arising from manual-coding procedures in which occupation codes followed the 5-digit format but corresponded to non-existent entries in the official KldB 2010 (2020 revision)—e.g., "84464", "81122", and "99999". These entries create impossible classification targets and would unjustly penalize system predictions, thereby undermining the validity of performance evaluation. Because this issue lies in format validity rather than semantic content, we classified it as preprocessing-addressable. Treatment involves cross-referencing each code against the authoritative taxonomy and systematically removing or flagging invalid entries, which ensures that only valid, realizable codes are included in the evaluation, preserving the integrity of the ground-truth labeling.

[10] https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

3.3.1.5 Systematic Coding Artifacts

We identified systematic coding artifacts arising from manual-coding procedures in which standardized placeholder codes (e.g., “0”, “-1”, “-2”, “-7”, “-9”, “-9990”) appeared repeatedly across multiple content fields. These entries carry no occupational information, may degrade data quality, and introduce noise that can interfere with system performance. Because this issue is syntactically distinct and consistently patterned, we classified it as preprocessing-addressable. Treatment involves identifying the full set of such artifacts through manual inspection and removing them via rule-based filtering, which can improve the signal-to-noise ratio in the dataset.

3.3.2 LLM-Addressable Preserved Issues

In contrast to the syntactic issues addressed through preprocessing, the following six issues exceed the scope of rule-based methods, requiring advanced semantic interpretation and contextual understanding. We intentionally preserved them to evaluate RAG-system robustness to linguistic variation and authentic survey-data complexity.

3.3.2.1 Expression Variability

We identified expression variability arising from survey-response patterns in which occupational descriptions vary significantly in input complexity (as defined in Section 1.3.3). On one end, responses consisted of minimal single-word entries—such as “Kassiererin” (cashier)—with empty or sparse task fields, representing low input complexity. On the other end, responses provided highly detailed descriptions—such as “Studienreferendarin am Gymnasium für die Fächer Biologie und Chemie” (student teacher at secondary school for biology and chemistry)—accompanied by comprehensive task lists like “Unterricht, Unterrichtsvorbereitung, Korrekturen, Elternkontakte” (teaching, lesson preparation, grading, parent contacts), representing high input complexity.

Low-input-complexity entries can introduce semantic ambiguity through insufficient detail, while high-input-complexity entries can introduce semantic ambiguity through contradictory, irrelevant, or redundant information (Section 1.1.2); both may challenge system performance. Because adaptive processing across this input-complexity spectrum requires contextual interpretation, semantic normalization, and knowledge of occupational domains beyond rule-based preprocessing, we classified this as LLM-addressable and intentionally preserved it. These capabilities can be supported in principle by LLMs, enabling authentic evaluation of system adaptability to varying input complexity.

3.3.2.2 Misplaced Task Descriptions

We identified misplaced task descriptions arising from survey-response patterns in which participants entered content types other than job titles into the title field. These included task-oriented phrases such as “Programmieren SEO” (programming SEO), activity descriptions like “Tätigkeit im Vertrieb und Marketing” (activity in sales and marketing), and even company names such as “BMW”. This structural misalignment

can complicate processing, as systems designed to interpret job titles encounter non-title content, thereby potentially impairing system performance. Because addressing this issue requires automatic content-type identification, field-relationship modeling, and semantic redistribution beyond rule-based preprocessing, we classified it as LLM-addressable and intentionally preserved it. These capabilities can be supported in principle by LLMs, enabling authentic assessment of system robustness to structural misalignment in naturalistic survey data.

3.3.2.3 Unknown Abbreviations

We identified unknown abbreviations arising from survey-response patterns, such as "GHS Lehrer" ("Grund- und Hauptschullehrer"—primary and lower secondary school teacher) and "LiV" ("Lehrer in Vorbereitung"—teacher in training). These forms introduce semantic ambiguity when context is limited, reflecting the terminological diversity and organizational jargon documented in Section 1.1.2, thereby potentially impairing system performance. Because addressing this issue requires knowledge of occupational abbreviations not available in our preprocessing resources, we classified it as LLM-addressable and intentionally preserved it. These capabilities—including abbreviation expansion via contextual interpretation and domain-specific knowledge—can be supported in principle by LLMs, enabling authentic assessment of system capability in interpreting occupational terminology within naturalistic language.

3.3.2.4 Inconsistent Capitalization Patterns

We identified inconsistent capitalization patterns arising from survey-response patterns, likely reflecting individual typing habits. Examples included "SOFTWAREENTWICKLER" (all uppercase), "softwareentwickler" (incorrect lowercase), and "Softwareentwickler" (correct German capitalization for software developer). Systems must recognize semantic equivalence across orthographic variations, and failure to do so may impair system performance. Because conventional case normalization risks compromising German orthographic correctness, and more sophisticated preprocessing would have introduced implementation complexity beyond our rule-based methods, we classified this issue as LLM-addressable and intentionally preserved it. Case-insensitive semantic interpretation can be supported in principle by LLMs, enabling authentic assessment of system robustness to orthographic variation without distorting linguistic norms.

3.3.2.5 Character-Encoding Degradation

We identified character-encoding degradation arising from data-processing artifacts in which German special characters (e.g., umlauts) were systematically replaced with question marks, transforming "Geschäftsführer" into "Gesch?ftsf?hrer". These corrupted forms result in information loss, likely hinder matching to standard occupational terminology, and thereby can impair system performance. Because the information loss is substantial—multiple characters map to a single question mark—and conventional techniques cannot reliably reconstruct them without external resources such as suitable dictionaries or character-mapping tables, we classified this issue as LLM-addressable and intentionally preserved it. Contextual understanding of corrupted representations can be supported in principle by LLMs, enabling authentic assessment of system resilience to data corruption and the ability to infer correct forms from context.

3.3.2.6 Semantic Duplicates

We identified semantic duplicates arising from survey-response patterns in which semantically equivalent job roles are expressed through different linguistic forms. Examples included job titles such as "Softwareentwickler" and "Programmierer" (programmer), paired with similar task descriptions that described comparable responsibilities. These variants reflect the terminological diversity documented in Section 1.1.2, requiring systems to recognize occupational equivalence despite linguistic differences. Inconsistent handling may impair system performance. Because detecting such semantic equivalence requires assessment of semantic similarity, contextual interpretation, and domain knowledge beyond syntactic matching, we classified this issue as LLM-addressable and intentionally preserved it. These capabilities can be supported in principle by LLMs, enabling authentic evaluation of system robustness to linguistic variation and the ability to generalize across synonymous expressions.

3.3.3 Inherent Dataset Characteristics

Beyond both preprocessing-addressable and LLM-addressable issues, the following two characteristics stem from intrinsic complexities in survey-data collection and manual coding processes that currently no known automated approach for survey-based occupation coding can resolve. We accepted them as unavoidable evaluation conditions that reflect authentic real-world scenarios.

3.3.3.1 Multi-Role Entries

We identified multi-role entries arising from survey-response patterns in which participants described multiple distinct occupational roles within a single response. Examples included "Verkäuferin und Lehrerin in einem Lerncenter" (saleswoman and teacher in a learning center) and "Personal- und Organisationsentwickler sowie Head of HR" (personnel and organizational developer as well as Head of Human Resources). These descriptions inherently represent multi-target content, creating fundamental classification ambiguity when evaluated under single-target frameworks. Such ambiguity can impair system performance, as systems designed for one-to-one mapping struggle to assign a single code to multi-faceted roles.

We classified this as an inherent dataset characteristic because resolving such entries requires expert authority to determine appropriate code assignments and access to the original coding rationale—factors that currently no automated system can readily replicate. Even trained human experts exhibit substantial disagreement in occupation coding, with inter-coder reliability reaching Cohen's Kappa values of 0.34 to 0.61 at the finest classification level (Section 1.1.2). Multi-role entries represent an extreme case of this inherent ambiguity, making automated resolution currently infeasible.

While LLM-based detection could flag such instances for review, this does not resolve the core issue of code-assignment authority. For example, splitting "saleswoman and teacher in a learning center" using suitable delimiters would produce entries lacking ground-truth codes. Conversely, interpreting "personnel and organizational developer as well as Head of Human Resources" as two distinct roles could be incorrect if

it reflects a single, integrated position. Thus, the decision must rest with human experts.

3.3.3.2 Multi-Label Classification Patterns

We identified multi-label classification patterns arising from manual coding procedures in which identical occupational descriptions received different KldB-code assignments across instances. Unlike multi-role entries where respondents describe multiple occupations per response, here expert coders assigned different codes to identical single-role descriptions. For example, "Kellner" (waiter) was assigned to both "63301" and "63302", and "Werkstudent" (working student) was assigned to "27104" and "72214". These inconsistencies reflect the low inter-coder reliability and inherent ambiguity in survey-based occupation coding, as documented in Section 1.1.2. As a result, they introduce artificial performance degradation when systems are penalized for correct predictions that differ from inconsistent ground-truth labels.

Resolving such patterns requires detailed knowledge of the original coding rationale and authoritative expert judgment. Although conventional preprocessing can easily detect these inconsistencies (Sections 3.5.3 and 3.6.2.2), determining which assignment is correct still requires occupational expertise—elements that currently no automated system can readily replicate. We therefore classified this as an inherent dataset characteristic that constitutes a fundamental constraint on evaluation.

3.3.4 Issue Classification Overview

Following the detailed characterization of individual issues, we consolidated all 13 data-quality problems across the DQAF's three dimensions to synthesize comprehensive treatment pathways. Our analysis identified issues spanning three origin categories—seven from survey-response patterns, four from manual-coding procedures, and two from data-processing artifacts—classified into three treatment classes: preprocessing-addressable, LLM-addressable preserved, and inherent dataset characteristics. This consolidated view reinforces clear boundaries between rule-based data preparation and required semantic capabilities reserved for RAG systems and their evaluation.

Table 3.2 consolidates five preprocessing-addressable issues that span all three origin categories and can be systematically resolved through deterministic rule-based methods. These issues represent syntactic and structural irregularities—such as non-informative participant responses and invalid taxonomy codes—that are eliminated prior to evaluation to establish consistent ground-truth conditions.

Table 3.3 consolidates six LLM-addressable preserved issues that exceed rule-based capabilities but remain addressable through advanced semantic processing. Five originate from survey-response patterns—ranging from expression variability to semantic duplicates—requiring contextual interpretation and semantic understanding alongside domain knowledge, while character-encoding degradation arises from processing artifacts and demands reconstructive interpretation. These issues are intentionally preserved to assess RAG-system resilience under real-world linguistic complexity, constituting core challenges—with five of six originating from participant-generated content—that likely determine system performance.

Table 3.2: Preprocessing-Addressable Issues in the Raw DZHW Survey Dataset. Five data-quality issues can be systematically resolved through deterministic rule-based methods prior to evaluation, establishing consistent ground-truth conditions. These issues span all three origin categories and are eliminated using deterministic techniques including pattern matching, format standardization, robust parsing, and cross-reference validation.

Data-Quality Issue	Issue Origin	Rule-Based Resolution Pathway
Non-Informative Participant Responses	Survey Patterns	Manual inspection to catalog placeholders, followed by pattern matching and systematic filtering.
Structural Data Corruption	Processing Artifacts	Robust parsing techniques (error-handling modes, custom delimiters) to accommodate malformed entries without content loss.
Inconsistent Code Formats	Manual Coding	Leading zero-padding for military codes and removal of invalid patterns to achieve 5-digit KldB 2010 standardization.
Invalid Taxonomy Codes	Manual Coding	Cross-reference validation against official KldB 2010 taxonomy; systematically remove or flag invalid entries.
Systematic Coding Artifacts	Manual Coding	Manual inspection to identify artifact patterns, followed by rule-based filtering.

Table 3.3: LLM-Addressable Preserved Issues in the Raw DZHW Survey Dataset. Six data-quality issues require advanced semantic processing and are intentionally preserved to assess RAG-system resilience under real-world conditions, with five of six originating from participant-generated content. These issues constitute core challenges that likely determine system performance and can be supported in principle by LLMs.

Data-Quality Issue	Issue Origin	Semantic Handling Pathway
Expression Variability	Survey Patterns	Adaptive processing across input-complexity levels via contextual interpretation, semantic normalization, and occupational domain knowledge.
Misplaced Task Descriptions	Survey Patterns	Content-type identification, field-relationship modeling, and semantic redistribution via contextual understanding and domain knowledge.
Unknown Abbreviations	Survey Patterns	Abbreviation expansion via contextual interpretation and domain-specific knowledge.
Inconsistent Capitalization Patterns	Survey Patterns	Case-insensitive semantic interpretation while preserving German orthographic conventions.
Character-Encoding Degradation	Processing Artifacts	Contextual understanding and reconstructive interpretation of corrupted German text.
Semantic Duplicates	Survey Patterns	Semantic similarity assessment, contextual interpretation, and domain knowledge for occupational-equivalence detection across linguistic variants.

Table 3.4 consolidates two inherent dataset characteristics that currently no known automated approach can resolve and must be accepted as unavoidable evaluation constraints. Multi-role entries stem from survey-response patterns in which participants describe multiple occupations per response, while multi-label classification patterns arise from manual-coding procedures producing inconsistent code assignments for identical descriptions. Both require expert judgment and access to original coding rationale, representing fundamental boundaries for reliable evaluation rather than targets for system improvement.

Table 3.4: Inherent Dataset Characteristics in the Raw DZHW Survey Dataset. Two data-quality characteristics currently cannot be resolved by any known automated approach and must be accepted as unavoidable constraints that represent fundamental boundaries for reliable evaluation. Both require expert judgment and access to original coding rationale, making them targets for expert intervention rather than system improvement.

Data-Quality Issue	Issue Origin	Expert-Based Resolution Pathway
Multi-Role Entries	Survey Patterns	Expert review to determine whether descriptions represent integrated or distinct roles, followed by authoritative code-assignment decisions based on original coding rationale.
Multi-Label Classification Patterns	Manual Coding	Expert consultation of coding guidelines and rationale to adjudicate between conflicting assignments, followed by authoritative determination of correct code.

These treatment pathways—rule-based resolution, semantic handling, and expert-based resolution—directly informed our preprocessing pipeline, RAG-system design, and evaluation methodology. The following section details the implementation of rule-based preprocessing and its quantitative outcomes for the five preprocessing-addressable issues.

3.4 Rule-Based Preprocessing Pipeline

Having classified data-quality issues into treatment classes, we designed a 6-step pipeline to address the five preprocessing-addressable issues identified in Section 3.3.1 and summarized in Table 3.2. Four targeted steps (Steps 1–4) resolved these five data-quality issues[11], while two additional steps (Steps 5–6) performed standard data-cleaning operations: removing empty instances and eliminating syntactic duplicates.

We applied this pipeline to the raw DZHW survey training dataset (Section 3.1) and measured its quantitative impact. In the following, we describe each step's operations and outcomes.

Step 1: Structural Corruption Resolution. We addressed structural corruption by implementing custom line-by-line parsing that treats double quotes as field boundaries, effectively bypassing the malformed quote-semicolon patterns that caused

[11]The five issues map to four pipeline steps because Step 4 addresses two distinct issues through a single coordinated operation.

CSV-parsing failures. This step restored reliable data loading for all subsequent operations.

Step 2: Code-Format Standardization. We applied a 2-stage approach to resolve code-format inconsistencies. First, we implemented automated leading zero-padding that converted 4-digit military codes to the required 5-digit KldB 2010 format, correcting 55 instances. Second, we applied pattern-based filtering that identified and removed entries with non–5-digit format violations, eliminating 193 instances.

Step 3: Invalid Taxonomy-Code Removal. We conducted systematic cross-reference validation against the official KldB 2010 taxonomy (2020 revision) to identify codes that followed the 5-digit format but corresponded to non-existent entries. This validation identified and removed 384 instances containing invalid taxonomy codes.

Step 4: Placeholder-Content Cleaning. We addressed both systematic coding artifacts and non-informative participant responses through coordinated pattern-based detection and standardization. We converted identified placeholder patterns—including internal codes such as "-1" and "-7", and reference phrases such as "siehe oben" (see above)—to empty strings, creating a uniform representation of missing content. This operation affected 28,228 instances across multiple fields: 39 job titles, 8,127 job-task 1 descriptions, and 20,062 job-task 2 descriptions.

Step 5: Empty-Instance Removal. We implemented content-availability filtering that removed 82 instances lacking substantive occupational information. This included 26 completely empty instances and 56 instances with empty job titles, ensuring that the evaluation dataset contains only entries with sufficient information for meaningful classification assessment.

Step 6: Syntactic-Duplicate Elimination. We applied deduplication procedures across all content fields to eliminate instances with identical text in all fields. This step removed 6,352 syntactic duplicates, reducing redundancy while preserving the unique occupational descriptions essential for robust and efficient evaluation.

Table 3.5 summarizes the six sequential steps, showing affected content fields, transformation examples, action types, and quantitative impact for each operation. Step 4 (placeholder-content cleaning) represents the highest-impact modification, affecting 28,228 instances (98.1% of all modifications). Step 6 (syntactic-duplicate elimination) represents the primary removal operation, affecting 6,352 instances (90.6% of all removals).

This systematic 6-step approach successfully resolved the five preprocessing-addressable issues, establishing consistent ground-truth conditions for evaluation. The six pipeline steps reduced the dataset from 52,542 to 45,531 instances (13.3% reduction). This reduction reflects targeted removal of unreliable and redundant entries rather than arbitrary filtering, yielding a preprocessed dataset that serves as the foundation for constructing four evaluation datasets (Section 3.6) tailored to our RAG-system experiments. The following section analyzes the characteristics of this preprocessed dataset.

Table 3.5: 6-Step Rule-Based Preprocessing Pipeline for the DZHW Survey Data. Step 4 (placeholder-content cleaning) affected 28,228 instances (98.1% of modifications); Step 6 (syntactic-duplicate elimination) affected 6,352 instances (90.6% of removals). Action types: M = Modified, R = Removed, NQ = Not Quantified (structural fixes without instance-level counts).

Preprocessing Step	Affected Field(s)	Example	Action	Count
1. Structural Corruption Resolution	All	Custom CSV parsing	M	NQ
2. Code-Format Standardization	Code	"1402" → "01402"; "-1" → *removed*	M/R	55/193
3. Invalid Taxonomy-Code Removal	Code	"84464" → *removed*	R	384
4. Placeholder-Content Cleaning	All	"-1"; "s.o." → ""	M	28,228
5. Empty-Instance Removal	All	*empty* → *removed*	R	82
6. Syntactic-Duplicate Elimination	All	*duplicates* → *removed*	R	6,352
Instances modified total:				**28,283**
Instances removed total:				**7,011**

3.5 Preprocessed Dataset Analysis

Having systematically resolved the five preprocessing-addressable issues through our 6-step pipeline, we analyze the resulting preprocessed dataset to inform the construction of evaluation datasets. Although preprocessing reduced the raw dataset from 52,542 to 45,531 instances (13.3% reduction) and resolved syntactic irregularities, the remaining instances still exceed our computational capacity for RAG-system evaluation due to expensive LLM-based inference operations under local hardware constraints (Appendix A.2.1). We must therefore construct smaller, strategically sampled evaluation datasets that enable meaningful assessment while maintaining taxonomic representativeness and authentic data-quality challenges.

This section addresses three analytical objectives that inform our sampling strategy, then synthesizes the structural challenges and data-quality characteristics that shape dataset construction. First, we characterize the distribution of KldB codes to assess class balance and identify potential evaluation biases (Section 3.5.1). Second, we quantify occupational code coverage to determine the maximum achievable taxonomic breadth given the preprocessed dataset's coverage limitations (Section 3.5.2). Third, we measure multi-label classification prevalence to assess whether evaluation results could be confounded by legitimate annotation disagreement (Section 3.5.3). Finally, we consolidate these findings to identify the primary structural challenges and preserved data-quality characteristics that constrain and motivate our dataset-construction approach (Section 3.5.4). We assume that the six LLM-addressable issues and two inherent dataset characteristics remain proportionally distributed in the preprocessed dataset, a reasonable simplification that avoids further analysis.

3.5.1 Class Imbalance and Distribution Patterns

Distributional analysis of 1-digit KldB codes reveals severe class imbalance with a pronounced long-tail pattern (Figure 3.1). Codes beginning with 8 (health and education) and 7 (business and administration) dominate the preprocessed dataset, comprising 56.3% of all instances with 15,327 and 10,304 occurrences, respectively. In

contrast, codes beginning with 0 (military) and 1 (agriculture and forestry) represent only 2.1%, with 47 and 893 occurrences. This severe skew reflects the real-world composition of survey participants—predominantly graduates and school leavers transitioning into professional roles—where education, healthcare, and business occupations are substantially over-represented while military and agricultural occupations remain under-represented.

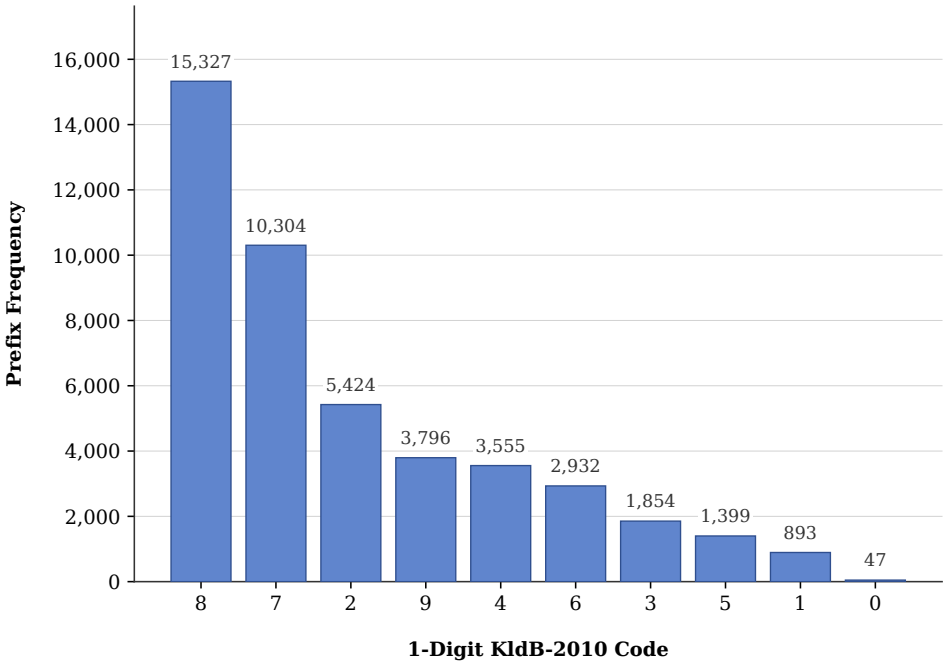


Figure 3.1: KldB 2010 Code Distribution in the Preprocessed DZHW Dataset. Codes beginning with 8 (health and education) and 7 (business and administration) dominate with 56.3% of instances (15,327 and 10,304 occurrences); codes 0 (military) and 1 (agriculture and forestry) represent 2.1% (47 and 893 occurrences). Long-tail distribution ordered by 1-digit prefix frequency, based on 45,531 instances across 10 unique 1-digit KldB 2010 codes.

Such extreme class imbalance typically necessitates specialized evaluation metrics to prevent evaluation bias favoring over-represented categories. However, a sampling strategy that selects one instance per KldB code would fundamentally eliminate this imbalance by ensuring equal representation across codes. Under such a design, specialized imbalance-handling metrics would become unnecessary, allowing standard evaluation metrics (Section 5.1.3) to provide unbiased performance assessment.

3.5.2 Taxonomy Coverage and Code Concentration

The preprocessed dataset demonstrates comprehensive coverage of the KldB 2010 taxonomy at higher levels while maintaining substantial breadth at deeper levels (Table 3.6). Coverage is complete at the first two hierarchical levels. Coverage decreases progressively from 98.6% at the 3-digit level to 89.2% at the 4-digit level and 83.7% at the 5-digit level (1,088 of 1,300 occupational types). While 16.3% of possible 5-digit codes remain absent—a coverage gap that cannot be closed

without collecting new survey data targeting missing categories—the dataset spans 1,088 distinct occupations, providing substantial taxonomic breadth for evaluation.

Table 3.6: KldB 2010 Code Coverage, Frequencies in the Preprocessed DZHW Dataset. Coverage decreases from 100% at first two levels to 83.7% at 5-digit level (1,088/1,300 codes); three most frequent codes ("84304", "84124", "83124") account for 14.4% of 45,531 instances.

Level	Coverage	1st		2nd		3rd	
		Code	Count	Code	Count	Code	Count
1-Digit	10/10 (100%)	**8**	15,327	**7**	10,304	**2**	5,424
2-Digit	37/37 (100%)	**84**	7,488	**71**	5,584	**81**	4,317
3-Digit	142/144 (98.6%)	**841**	3,306	**843**	2,956	**831**	2,743
4-Digit	626/702 (89.2%)	**8430**	2,947	**8412**	2,034	**8312**	1,648
5-Digit	1,088/1,300 (83.7%)	**84304**	2,947	**84124**	2,034	**83124**	1,594

Beyond coverage breadth, significant code concentration emerges at the finest granularity (Table 3.6). The three most frequent 5-digit codes account for 14.4% of all instances. Code "84304" denotes higher-education teaching and research (e.g., "Honorarprofessor"—honorary professor) with 2,947 instances. Code "84124" denotes secondary-education teachers (e.g., "Lehrer"—teacher) with 2,034 instances. Code "83124" denotes social work and pedagogy (e.g., "Sozialarbeiter"—social worker) with 1,594 instances. This concentration reinforces the participant-composition bias observed at the 1-digit level, where education and social-service professions dominate the dataset.

3.5.3 Multi-Label Classification Prevalence

We identified 847 instances (1.86% of the preprocessed dataset) in which identical occupational descriptions received different code assignments from expert coders, quantifying the prevalence of multi-label classification patterns documented in Section 3.3.3.2. These patterns represent legitimate annotation ambiguity where multiple codes may be defensible for identical input text, potentially introducing evaluation noise when RAG systems retrieve valid alternative codes that are penalized as incorrect. However, the limited prevalence suggests this ambiguity would affect a negligible proportion of evaluation instances. Under a stratified sampling strategy that selects one instance per code, such patterns would be naturally distributed across separate strata without requiring specialized handling. We therefore treated multi-label patterns as a tolerable level of annotation inconsistency that introduces marginal evaluation noise reflecting documented inter-coder disagreement.

3.5.4 Dataset Complexity and Remaining Challenges

Synthesizing the findings from our code distribution and coverage analyses, the preprocessed dataset presents two primary structural challenges—alongside its size—for our RAG-system evaluation. First, severe class imbalance (where the top three codes alone comprise 14.4% of instances) would introduce evaluation bias favoring over-represented categories under naive sampling approaches. A stratified sampling

strategy that selects one instance per code could address this challenge by ensuring equal representation across codes, thereby eliminating imbalance while maximizing taxonomic breadth up to the 1,088 codes present in the preprocessed dataset.

Second, this stratified sampling approach encounters a constraint when combined with the need to assess system robustness across the expression-variability spectrum observed in the raw survey data. The preprocessed dataset imposes a coverage ceiling of 1,088 codes (83.7% of the 1,300 codes in the KldB 2010), which represents substantial but incomplete taxonomic breadth. Incorporating input-complexity variation through stratified sampling would require constraining strata to codes with sufficient within-stratum variability, which could further reduce achievable code coverage below this ceiling. Evaluation datasets must therefore balance the trade-off between maximizing taxonomic breadth and inducing controlled input-complexity conditions, with some datasets prioritizing one objective over the other.

Beyond these structural challenges, the preprocessed dataset is expected to retain authentic data-quality characteristics that remain relevant for RAG-system evaluation. We intentionally preserved six LLM-addressable issues and two inherent dataset characteristics, which our DQAF systematically distinguished from preprocessing-addressable syntactic irregularities. Expression variability (Section 3.3.2.1) and unknown abbreviations (Section 3.3.2.3) exemplify the brevity and linguistic sparsity documented by Safikhani et al. [2023], who characterized the DZHW dataset as "complex and challenging" because survey participants "have only a vague idea of their future jobs", resulting in brief descriptions with "limited linguistic features" in a dataset that "has not been extensively curated".

While Safikhani et al. [2023] improved prediction performance by removing 935 entries identified as having incorrect code assignments from their 52,807-instance dataset, our approach extends their work through structured treatment pathways across 13 distinct issues spanning three origin categories (Section 3.2). Rather than focusing solely on the removal of ground-truth coding errors, we removed 7,011 entries (13.3%) and corrected syntactic irregularities in 28,283 entries (53.8%) across five preprocessing-addressable issues to ensure ground-truth validity. The following section describes how we leverage these analytical findings to construct four evaluation datasets that balance computational feasibility with taxonomic representativeness and authentic linguistic complexity.

3.6 Evaluation Datasets

To address the structural challenges identified in our preprocessed dataset analysis—severe class imbalance and the trade-off between taxonomic breadth and input-complexity variation—we constructed four complementary evaluation datasets from the preprocessed dataset using stratified sampling with KldB codes as strata. This approach eliminates class imbalance by selecting one instance per code while enabling controlled assessment of RAG-system robustness across varying input-complexity levels.

The four datasets balance competing objectives: one dataset (random all dataset, $n = 1{,}088$) maximizes taxonomic breadth by including all codes present in the

preprocessed dataset, while three datasets (longest job title dataset, shortest job title dataset, random valid dataset, each $n = 856$) induce controlled input-complexity conditions—ranging from low to high—by selecting from codes with sufficient job-title length variation.

The subsections below detail the construction methodology (Section 3.6.1), including the stratification procedure, the resulting dataset definitions and their taxonomic coverage, and the validity constraints that bound interpretation. We then validate dataset suitability (Section 3.6.2) by quantifying input-complexity variation, measuring the extent of quantifiable data-quality issues, and discussing implications of non-quantifiable issues for evaluation validity.

3.6.1 Construction Methodology

This section details the stratified sampling procedure we applied to construct the four evaluation datasets, including the practical constraints that shaped our design, the proxy measure we used to operationalize input complexity, the stratification outcomes that define each dataset's analytical role, and the validity considerations that bound interpretation.

3.6.1.1 Constraints and Design Rationale

Two practical constraints necessitated a reduction in dataset size and informed our stratified sampling design. First, multi-stage LLM inference in our RAG pipelines is computationally intensive, typically requiring substantial cloud or cluster resources. Second, DZHW data-privacy rules constrained evaluation to local machines with limited capacity (Appendix A.2.1). Together, these constraints required reducing the preprocessed dataset (PPD) from 45,531 instances to approximately 1,000 instances per evaluation dataset to enable processing within local resource budgets.

We addressed these constraints through stratified sampling that uses KldB codes as strata and selects one instance per stratum. This design directly addresses the severe class imbalance identified in Section 3.5.1 by ensuring equal representation across codes, eliminating the need for specialized evaluation metrics for handling imbalance. The one-instance-per-code selection additionally maximizes achievable code coverage within the computational budget, yielding small, representative, privacy-compliant datasets suitable for evaluation.

Beyond addressing class imbalance, we designed the datasets to enable controlled assessment of RAG-system robustness across varying input-complexity conditions. This required distinguishing two strata types: (i) *constrained strata* containing codes where instances exhibit sufficient within-stratum variation in job-title length to enable meaningful comparisons between low and high complexity, and (ii) *unconstrained strata* containing all codes present in the preprocessed dataset to preserve complete taxonomic coverage. This 2-tier design enables multiple analytical perspectives—some datasets prioritizing input-complexity variation, others prioritizing taxonomic breadth—compared with using a single random dataset of similar aggregate size.

3.6.1.2 Input-Complexity Proxy and Stratification Criteria

To distinguish constrained from unconstrained strata, we operationalized input complexity using job-title character length as a practical proxy. Two choices defined this proxy: (i) measuring character length rather than semantic complexity scores to maintain tractability, and (ii) applying the measure to job-title fields rather than job-task descriptions to align with RAG' prioritized inputs. This proxy does not fully capture semantic richness, domain-specific terminology, or classification difficulty, but it provides a directly measurable indicator of input complexity.

We applied a simple threshold to identify constrained strata: for each KldB code stratum, we computed job-title lengths for all instances in that stratum, then classified the stratum as constrained if the difference between its shortest and longest job titles (i.e., the within-stratum length range) reached at least 10 characters. This pragmatic threshold balances broad code coverage with measurable input-complexity differences while keeping the criterion simple and reproducible. More elaborate threshold-selection methods, such as elbow techniques, were not pursued because they fall outside our scope and would introduce avoidable complexity.

3.6.1.3 Stratification Outcomes and Dataset Definitions

Applying the 10-character threshold to the 45,531 preprocessed instances organized into 1,088 code-based strata, we identified 856 constrained strata suitable for input-complexity comparisons and 232 unconstrained strata with insufficient variation. This stratification produced four evaluation datasets, each serving a distinct analytical role:

1. Longest job title dataset (LJT): Contains the instance with the longest job title from each of the 856 constrained strata, representing highest input-complexity conditions for robustness testing.
2. Shortest job title dataset (SJT): Contains the instance with the shortest job title from each of the 856 constrained strata, representing lowest input-complexity conditions for robustness testing.
3. Random valid dataset (RVD): Contains one randomly selected instance from each of the 856 constrained strata, providing a baseline for input-complexity comparisons.
4. Random all dataset (RAD): Contains one randomly selected instance from each of the 1,088 code strata, ensuring complete taxonomic coverage of the preprocessed dataset.

These datasets are not mutually exclusive—identical instances can appear in multiple datasets depending on random selection and stratum composition. All random selections used fixed seeds (Appendix A.2.2) to ensure reproducibility.

Occupational code coverage reflects the intended design trade-offs. As Table 3.7 shows, RAD achieves complete preprocessed coverage (1,088 codes), while the three constrained datasets retain 856 codes, reflecting the 10-character threshold requirement for input-complexity variation.

Table 3.7: KldB 2010 Code Coverage Across the Four Evaluation Datasets. RAD achieves complete preprocessed coverage (1,088 codes, 83.7% of official taxonomy); constrained datasets retain 78.7% of preprocessed coverage (856 codes, 65.8% of official taxonomy), reflecting the trade-off between taxonomic breadth and induced input-complexity variation. Coverage reported relative to the preprocessed dataset (1,088 codes) and the official KldB 2010 taxonomy (1,300 codes).

Coverage Metric	LJT	SJT	RVD	RAD	PPD
Unique KldB 2010 codes	856	856	856	1,088	1,088
Coverage vs. preprocessed (%)	78.7	78.7	78.7	100.0	100.0
Coverage vs. official KldB (%)	65.8	65.8	65.8	83.7	83.7

3.6.1.4 Validity Considerations

The construction methodology faces two validity constraints that bound interpretation. First, stratified sampling with one instance per KldB code yields exactly one observation per occupational type (5-digit code) in each dataset, precluding within-code statistical analyses at the finest taxonomic granularity. The multi-dataset design partially mitigates this limitation: for the 856 constrained codes, aggregating observations across all four datasets provides up to four instances per code, enabling cross-dataset analysis of systematic performance patterns and input-complexity effects.

Second, findings depend on the construct validity of job-title character length as a proxy for input complexity and on the external validity of the preprocessed DZHW dataset, which covers 83.7% of the KldB 2010 taxonomy (Section 3.5.2). While this coverage ceiling constrains all evaluation datasets equally, the constrained datasets face an additional reduction to 65.8% of the taxonomy due to the 10-character threshold requirement, whereas RAD preserves the full 83.7% preprocessed coverage (Table 3.7). Consequently, findings generalize to different fractions of the occupational landscape depending on which dataset is used.

3.6.2 Dataset Characteristics and Suitability Assessment

Having defined the construction methodology, we validate whether the four evaluation datasets exhibit the intended input-complexity variation and preserve authentic data-quality issues relevant to realistic RAG evaluation. We quantify input-complexity variation across datasets through job-title and task field analysis (Section 3.6.2.1), measure the extent of quantifiable data-quality issues (Section 3.6.2.2), and discuss implications of non-quantifiable issues for realistic assessment (Section 3.6.2.3).

3.6.2.1 Input-Complexity Analysis

This section tests whether the four evaluation datasets exhibit the intended spread in input complexity across job-title and task fields. We examine (i) job-title length distributions and (ii) job-task field occupancy to verify effects beyond titles. By design, the *extreme datasets* (LJT, SJT) bracket the spectrum of input complexity, whereas the *baseline datasets* (RVD, RAD) mirror natural-language norms of the PPD.

Figure 3.2 displays job-title character-length distributions for all evaluation datasets and the preprocessed dataset. Additionally, Table 3.8 reports the corresponding descriptive statistics for the plotted distributions. The extreme datasets show pronounced separation of nearly 50 characters in median length (LJT high at 58.0, SJT low at 9.0). The baseline datasets align with the preprocessed distribution (medians around 21, similar quartiles, means near 26), indicating that our stratified construction achieved both separation and realism. These patterns support controlled evaluation across low and high input complexity while preserving baseline comparability to the preprocessed dataset.

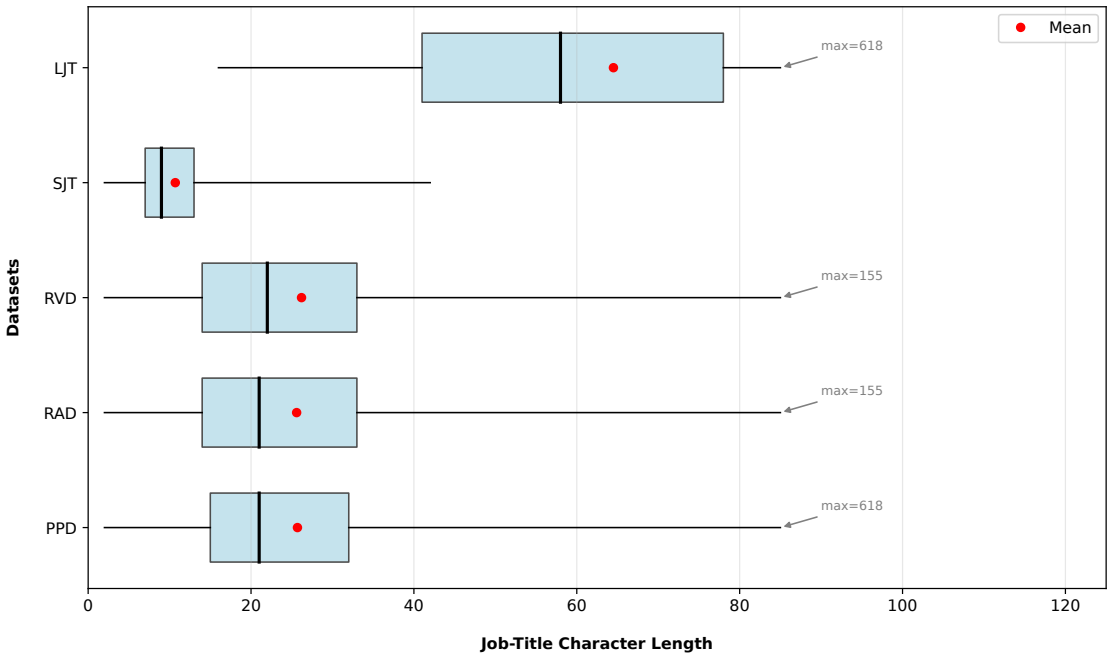


Figure 3.2: Job-Title Length Distributions Across the Four Evaluation Datasets. Extreme datasets show strong contrast (LJT median 58.0 vs. SJT median 9.0 characters, 49-character difference); baseline datasets align with PPD (medians around 21, means around 26). The plot shows box plots of job-title character-length distributions for the four evaluation datasets and preprocessed dataset. X-axis clipped at 85 characters with maxima annotated to accommodate long-tail values.

Table 3.9 reports job-task completion rates for all evaluation datasets and the preprocessed dataset, measured as the share of instances with at least 15 characters in task fields 1 and 2. The higher 15-character cutoff serves as a pragmatic proxy for input complexity in task descriptions, which are typically more verbose than job titles. The extreme datasets bracket the range—LJT highest and SJT lowest in both task fields—while the baseline datasets lie between and near the PPD. Notably, completion rates rise with job-title length, suggesting that respondents who write longer job titles also tend to provide more detailed task descriptions. These results validate job-title character length as a proxy for input complexity, as longer titles correlate with more detailed task descriptions. Combined with the title-length distributions above, these patterns enable controlled evaluation across low and high input-complexity conditions while preserving realistic variation.

Table 3.8: Job-Title Length Statistics Across the Four Evaluation Datasets. Extreme datasets show strong contrast (LJT median 58.0 vs. SJT median 9.0 characters, 49-character difference); baseline datasets align with PPD (medians around 21, means around 26). Rows show descriptive statistics (minimum, quartiles, median, maximum, interquartile range, mean, sample size) of job-title character-length distributions for the four evaluation datasets and preprocessed dataset.

Dataset	Min	Q1	Median	Q3	Max	IQR	Mean	N
LJT	16	41.0	58.0	78.0	618	37.0	64.5	856
SJT	2	7.0	9.0	13.0	42	6.0	10.7	856
RVD	2	14.0	22.0	33.0	155	19.0	26.2	856
RAD	2	14.0	21.0	33.0	155	19.0	25.6	1,088
PPD	2	15.0	21.0	32.0	618	17.0	25.7	45,531

Table 3.9: Job-Task Completion Rates Across the Four Evaluation Datasets. LJT shows highest completion rates (73.5% Task 1, 47.0% Task 2); SJT shows lowest (62.3%, 31.3%); baseline datasets lie between and near PPD values. Rows show completion rates for job-task fields 1 and 2 across the four evaluation datasets and PPD. Completion measured as percentage of instances with at least 15 characters, serving as a pragmatic proxy for input complexity in job-task descriptions.

Completion Metric	LJT	SJT	RVD	RAD	PPD
Task 1 completion (%, ≥15 chars)	73.5	62.3	66.9	65.6	73.0
Task 2 completion (%, ≥15 chars)	47.0	31.3	39.0	36.6	46.0

3.6.2.2 Preservation of Quantifiable Data-Quality Issues

This section examines whether and to what extent quantifiable data-quality issues remain in the four evaluation datasets to ensure realistic RAG evaluation challenges. We (i) identified which issues are quantifiable under our computational constraints, (ii) operationalized detection heuristics where needed, and (iii) measured their extent across datasets.

Of the 13 data-quality issues identified in Section 3.3.4, five were resolved through rule-based preprocessing (Section 3.4), leaving eight issues in the preprocessed dataset: six LLM-addressable issues intentionally preserved for evaluation and two inherent dataset characteristics. Four of these eight are quantifiable under our computational constraints: character-encoding degradation, inconsistent capitalization patterns, expression variability, and multi-label classification patterns. The remaining four require sophisticated detection techniques or extensive manual annotation and are therefore out of scope. We now detail the operationalization and measurement of each quantifiable issue.

Expression Variability

We operationalized expression variability using the input-complexity distributions already established in Section 3.6.2.1. Job-title length distributions and task-field completion rates capture variation in expression across datasets: LJT exhibits longer

titles and higher task-field completion (73.5% Task 1, 47.0% Task 2), potentially providing greater semantic density for matching with KldB 2010 documents, while SJT exhibits shorter titles and lower completion (62.3%, 31.3%), testing robustness under sparse input conditions. Baseline datasets mirror preprocessed norms. This proxy does not capture all dimensions of expression variability but provides a measurable indicator under our constraints.

Multi-Label Classification Patterns

We quantified multi-label patterns by grouping instances with identical content across all fields (job title and tasks 1–2) and counting unique KldB codes per content group. Groups with multiple codes indicate multi-label ambiguity. Stratified sampling does not eliminate these patterns because it selects one instance per code without considering content duplication. Measured at the instance level, multi-label rates are: LJT 0/856 (0.00%), SJT 33/856 (3.86%), RVD 2/856 (0.23%), RAD 4/1,088 (0.37%), compared with 1.86% in the PPD (Section 3.5.3).

The extreme datasets diverge substantially: SJT exhibits the highest rate (3.86%), while LJT shows no affected instances. This pattern aligns with the input-complexity findings in Section 3.6.2.1: shorter job titles and fewer completed task fields increase the likelihood of identical content across instances, and when combined with annotator disagreement, this yields higher multi-label rates. Baseline datasets show rates below the PPD, likely reflecting stratified sampling's distributional effects. For RAG evaluation, SJT results require careful interpretation because multi-label ambiguity may depress performance metrics if models predict valid but non-primary labels.

Character-Encoding Degradation

We detected character-encoding degradation—systematic replacement of German umlauts with question marks (Section 3.3.2.5)—by scanning all content fields for question-mark presence. If any field contains a question mark, the instance is flagged as corrupted. This heuristic captures many corrupted umlauts while accepting some false positives from question marks used as punctuation, a trade-off necessary under our constraints.

Inconsistent Capitalization Patterns

We detected capitalization inconsistencies—varied typing preferences producing multiple formats for identical concepts (Section 3.3.2.4)—by identifying all-uppercase text exceeding three characters across all content fields. This threshold excludes standard German three-character initialisms ("Initialwörter") [Kobler-Trill, 1994] while capturing longer inconsistencies. If any field exceeds the threshold, the instance is flagged. This approach balances precision and recall, occasionally including longer acronyms (e.g., "DSGVO") as acceptable false positives.

Prevalence Across Datasets

Table 3.10 reports the extent of encoding and capitalization issues across datasets and content fields. Character-encoding degradation affects 2.5% to 3.9% of evaluation-dataset instances, slightly below the PPD rate of 4.7%. Capitalization inconsistencies

are substantially higher in evaluation datasets (40.8% to 51.8%) than in the PPD (25.5%). The elevated capitalization rates likely reflect stratified sampling effects: the PPD contains many instances concentrated in few codes (Section 3.5.1), and sampling one instance per code shifts the distribution toward codes with fewer instances, which may exhibit higher capitalization inconsistency rates. Together, these findings confirm that quantifiable data-quality issues persist at realistic rates, ensuring that the evaluation datasets remain sufficiently challenging for RAG assessment. Section 3.6.2.3 discusses the four non-quantifiable issues and their implications for evaluation validity.

Table 3.10: Encoding, Capitalization Issues Across the Four Evaluation Datasets. Character-encoding degradation affects 2.5–3.9% of evaluation-dataset instances, slightly below PPD (4.7%); capitalization inconsistencies affect 40.8–51.8%, substantially above PPD (25.5%). Rows show counts and percentages for both quality issues across datasets, broken down by content field (Job Title, Tasks 1 and 2) with combined instance-level totals. Character-encoding degradation measured via question-mark presence; capitalization inconsistency via all-uppercase text exceeding 3 characters.

Quality Issue	Content Field	LJT		SJT		RVD		RAD		PPD	
Character-encoding degradation (presence of '?'):											
	Job Title	10	(1.2%)	8	(0.9%)	8	(0.9%)	10	(0.9%)	808	(1.8%)
	Job Tasks 1	8	(0.9%)	17	(2.0%)	11	(1.3%)	14	(1.3%)	1,089	(2.4%)
	Job Tasks 2	11	(1.3%)	14	(1.6%)	8	(0.9%)	11	(1.0%)	1,117	(2.5%)
	Combined	**22**	**(2.6%)**	**33**	**(3.9%)**	**21**	**(2.5%)**	**27**	**(2.5%)**	**2,139**	**(4.7%)**
Inconsistent capitalization (all-uppercase >3 chars):											
	Job Title	345	(40.3%)	427	(49.9%)	383	(44.7%)	519	(47.7%)	11,322	(24.9%)
	Job Tasks 1	207	(24.2%)	292	(34.1%)	247	(28.9%)	325	(29.9%)	8,045	(17.7%)
	Job Tasks 2	3	(0.4%)	2	(0.2%)	0	(0.0%)	0	(0.0%)	124	(0.3%)
	Combined	**349**	**(40.8%)**	**443**	**(51.8%)**	**387**	**(45.2%)**	**523**	**(48.1%)**	**11,604**	**(25.5%)**

3.6.2.3 Implications of Non-Quantifiable Data-Quality Issues

Four data-quality issues identified in the raw survey data (Section 3.3.4)—misplaced task descriptions, unknown abbreviations, semantic duplicates, and multi-role entries—were intentionally preserved through preprocessing and should therefore persist in the evaluation datasets at rates proportional to the PPD. Because stratified sampling selects instances based on KldB codes without filtering semantic content, these issues should survive the construction process. However, unlike the issues addressed in Section 3.6.2.2, these four require sophisticated NLP techniques or extensive manual annotation for reliable detection and cannot be quantified under our computational constraints. We reason qualitatively about their expected prevalence across datasets and discuss implications for evaluation validity, focusing on how input-complexity differences between LJT and SJT likely affect issue manifestation.

Reasoning Framework

Because we cannot measure these issues directly, we reason about their expected prevalence using input complexity as a proxy. The stratified construction produced

systematic differences: LJT contains longer job titles (median 58 characters) and higher task-field completion rates (73.5% Task 1), while SJT contains shorter titles (median 9 characters) and lower completion rates (62.3%). These differences create distinct conditions for issue manifestation: longer, content-rich entries provide more opportunities for certain issues to appear but also supply more contextual cues for resolution, whereas shorter entries constrain issue frequency but reduce disambiguating information when issues do occur. We consider each issue's likely behavior under these opposing conditions.

Content-Driven Issues

Three issues—misplaced task descriptions, unknown abbreviations, and semantic duplicates—likely exhibit higher absolute frequency in LJT due to greater content volume, but their impact on evaluation validity differs by issue type. Misplaced task descriptions (Section 3.3.2.2) occur when respondents enter task information in the job-title field. Longer LJT titles provide more space for such spillover, potentially blurring the field-semantic distinctions that our RAG approaches rely on. Unknown abbreviations (Section 3.3.2.3) are domain-specific short forms lacking contextual expansion. While longer LJT entries may contain more abbreviations in absolute terms, they also provide surrounding context that aids interpretation; shorter SJT entries may use abbreviations less frequently but offer minimal disambiguating cues when present. Semantic duplicates (Section 3.3.2.6) arise when distinct phrasings express identical occupational concepts. Content-rich LJT entries admit greater expressive variety and thus more opportunities for paraphrase, testing whether RAG systems recognize conceptual equivalence across surface forms.

Multi-Role Entries

Multi-role entries (Section 3.3.3.1) describe multiple occupational roles within a single instance but receive only one KldB code, representing an inherent dataset characteristic that no automated approach can fully resolve. We expect multi-role descriptions to surface more readily in longer, detailed LJT entries, where respondents have space to describe varied responsibilities, whereas SJT's brevity constrains explicit multi-role articulation. This issue challenges evaluation validity because models may return plausible codes for unlabeled secondary roles yet be scored as incorrect, making performance metrics conservative estimates of true capability.

Evaluation Validity Implications

The presence of these non-quantifiable issues affects evaluation validity in three ways. First, performance metrics represent lower bounds on true model capability: models may produce valid alternative classifications for multi-role entries or semantically equivalent predictions for duplicate expressions, yet be scored as incorrect under exact-match evaluation. Second, LJT and SJT test distinct challenge dimensions—content richness and field-boundary ambiguity versus sparse input and minimal context—rather than a single difficulty continuum, requiring careful comparative interpretation. Third, the baseline datasets (RVD, RAD) preserve the PPD's proportional issue distribution, providing reference points for distinguishing input-complexity effects

from construction artifacts. Together, these considerations frame result interpretation in subsequent analyses (Chapter 5): performance differences across datasets may reflect genuine robustness variation, but some portion likely stems from unmeasured issues affecting datasets differentially.

3.7 Summary and Contributions

This chapter establishes a systematic approach to prepare the raw DZHW survey dataset for RAG-system evaluation. We assessed data quality using our 3-dimensional DQAF, then preprocessed the dataset to resolve five syntactic issues while preserving six LLM-addressable challenges and two inherent characteristics. We constructed four evaluation datasets through stratified sampling and validated them for input-complexity variation and authentic data-quality preservation.

These datasets directly address the research questions' need for controlled input-complexity variation—a key dimension of information richness (Section 1.3.3, Section 1.3.4). The extreme datasets (LJT, SJT) enable evaluating LLM-based query rewriting under low and high input-complexity conditions. RVD provides an input-complexity baseline within the same 856-code set, while RAD preserves complete coverage across all 1,088 occupational codes for general benchmarking. LJT offers the most challenging evaluation setting through greater semantic density and potential field-boundary ambiguity; SJT tests performance under sparse contextual information where brief inputs provide minimal disambiguating cues.

This work contributes three principal advances:

- **Methodologically**, we developed a 3-dimensional DQAF that systematically distinguishes preprocessing-addressable issues from semantic challenges reserved for system evaluation. A 6-step preprocessing pipeline operationalizes this framework, establishing ground-truth validity while preserving authentic linguistic complexity.
- **Empirically**, we characterized the preprocessed dataset comprehensively: severe class imbalance, substantial taxonomic coverage (1,088 of 1,300 codes), and limited multi-label ambiguity (1.86%). We validated input-complexity separation across evaluation datasets, demonstrating median job-title lengths of 58 versus 9 characters for LJT versus SJT.
- **As resources**, we produced four stratified evaluation datasets (RAD, $n = 1{,}088$; LJT, SJT, RVD, each $n = 856$) that balance computational feasibility with taxonomic representativeness and authentic data-quality challenges. These datasets enable controlled assessment of RAG-system robustness under realistic survey-data conditions. Though DZHW-internal due to data-privacy constraints, our preprocessing procedures and stratification methods are fully documented for reproduction on independent datasets.

4. System Implementation

This chapter documents the implementation of the two RAG-based occupation-coding systems evaluated in this thesis: Naive-RAG and QR-RAG. Both systems operate on the preprocessed DZHW survey data described in Chapter 3 and query a shared knowledge base constructed from KldB 2010 documentation. The chapter proceeds systematically from general to specific. We first present the system design overview (Section 4.1), justifying our selection of vector-based RAG, contrasting the two pipeline designs, and explaining how computational constraints shaped implementation decisions. We then describe the construction of the shared knowledge base that both systems query (Section 4.2), followed by the pipeline components common to both approaches—retrieval and classification stages that process queries and assign occupation codes (Section 4.3). Finally, we detail the specific implementations: Naive-RAG's direct query formulation from original survey data (Section 4.4) and QR-RAG's three-stage query-rewriting pipeline, its concurrent batch-prompting architecture, and its extended set of query formulations (Section 4.5).

4.1 System Design Overview

This section establishes the foundational design decisions that shaped both RAG implementations, justifying our selection of vector-based RAG, comparing the two pipeline designs, and explaining how computational constraints influenced implementation choices.

4.1.1 RAG-Paradigm Selection

We selected a vector-based RAG paradigm for both approaches, wherein queries and knowledge-base documents are embedded into dense vector representations and retrieval proceeds via semantic similarity search. This section contextualizes that selection within contemporary RAG variants and provides implementation rationale.

Modern RAG systems employ diverse retrieval strategies, each with distinct strengths and implementation requirements. Vector-based RAG embeds queries and documents into dense semantic spaces using embedding models, then retrieves candidates via

semantic search using metrics like cosine similarity. This paradigm offers strong semantic matching across linguistic variation, multilingual support, and straightforward implementation with established vector databases. Performance depends heavily on embedding quality, and exact lexical matches may be missed. Hybrid RAG combines sparse lexical methods (e.g., BM25) with dense semantic retrieval [Bruch, 2024], fusing keyword matching with semantic similarity. These systems balance exact-match precision with semantic robustness but require careful fusion-strategy tuning and increase implementation complexity. Graph-based RAG constructs knowledge graphs representing entities and relationships, retrieves via graph traversal, and integrates retrieved subgraphs into generation context [Evangelista et al., 2026; Peng et al., 2025]. This approach provides explicit entity modeling, multi-hop reasoning support, and interpretable retrieval paths. However, the primary limitation lies in substantial construction overhead: even when a structured source taxonomy exists, building a traversable knowledge graph requires dedicated schema design, relationship modeling beyond the source hierarchy, and graph retrieval infrastructure.

We selected vector-based RAG for three reasons aligned with our research objectives and constraints. First, occupation coding inherently requires semantic interpretation. Survey respondents express job roles using diverse terminology, abbreviations, and informal descriptions that rarely match formal KldB descriptions exactly. Dense vector embeddings capture semantic similarity across such linguistic variation, enabling matching despite surface-form differences—a capability essential for bridging the semantic gap (Section 1.1.2). Second, vector-based RAG represents the predominant paradigm in recent research, providing well-established baselines for contextualization. This established foundation facilitates interpretation of findings within the broader RAG literature and enables meaningful comparisons with related occupation-coding studies. Third, implementation feasibility favored vector-based RAG. The paradigm requires only authoritative text and pre-trained EMs, both readily available, enabling focus on core research questions. Hybrid approaches demand extensive fusion-strategy optimization across multiple retrieval methods, while graph-based systems require constructing a purpose-built knowledge graph over KldB's 5-level, 1,300-category hierarchy—both substantially exceeding our implementation scope.

4.1.2 Pipeline-Design Overview

The two approaches differ in a single design dimension: QR-RAG prepends an LLM-based query-rewriting stage to the Naive-RAG pipeline, transforming original survey data before query formulation. Naive-RAG processes original survey data through three sequential stages. First, job titles and task descriptions are composed into query strings using one of three predefined formulation strategies: Title-Only, Tasks-Only, or Title+Tasks. Each strategy selects specific data fields from the original survey data. Second, formulated queries are embedded using dual-encoder EMs and matched against pre-embedded task descriptions in the vector store via semantic similarity search, retrieving top-k candidates with their corresponding KldB codes. Third, retrieved candidates and their KldB codes augment the LLM classification prompt requesting occupation-code assignment. This approach tests whether participant language in its original form—often informal, abbreviated, or incomplete—suffices for accurate occupation coding without LLM-based query rewriting.

QR-RAG inserts a query-rewriting stage before query formulation, extending the pipeline to four stages. The additional stage transforms original job titles and task descriptions through a three-stage LLM-based query-rewriting pipeline designed to improve survey-data quality and query–document alignment, producing cleaned and enhanced data including hypothetical job tasks optimized for retrieval against task-oriented knowledge-base documents. The rewritten data then enters query formulation, where one of four strategies is applied: three strategies parallel to those in Naive-RAG, plus the Schema-Data strategy exclusive to QR-RAG. Retrieval and classification proceed identically to Naive-RAG. This approach tests whether LLM-based query rewriting improves retrieval effectiveness and downstream classification performance.

As illustrated in Figure 4.1, the shared retrieval and classification stages employ the same models and logic across both approaches, deliberately isolating query preparation as the sole experimental variable. The knowledge base remains constant across all experiments. The optimal EM and LLM-based classifier are likewise shared, both selected through component-level experiments documented in Sections 5.2 and 5.3.

4.1.3 Computational Constraints and Implementation Decisions

Hardware limitations and data-privacy requirements shaped critical implementation decisions throughout both approaches. The experimental workstation comprised dual NVIDIA GeForce RTX 3060 graphics processing units (GPUs) with 12 GiB video random access memory (VRAM) each (24 GiB total), 128 GiB system RAM, and an AMD Ryzen 9 5950X processor (16 cores, 32 threads). While substantial for consumer hardware, this configuration falls below typical cloud or cluster resources used in large-scale LLM research. Additionally, DZHW data-privacy regulations mandated local processing on this workstation, precluding cloud-based inference services. These constraints necessitated both careful inference engine selection and pipeline optimizations to enable experimental configurations involving nine EMs and six QR-LLMs within available resources.

We selected `Ollama`[12] as our LLM inference engine for five key reasons: (i) developer-friendly design enabling rapid prototyping with plug-and-play model deployment, (ii) deployment supported on consumer-grade GPU hardware, (iii) Python ecosystem integration through `langchain-ollama`[13], (iv) built-in concurrent request handling, and (v) native think mode support for reasoning-capable models. These capabilities position Ollama as a practical choice compared to open-source alternatives like `vLLM` [Adeboye, 2025], which require substantially more infrastructure configuration expertise. Ollama's limitation to a curated model repository constrained our model selection to available variants, as documented in Sections 5.1.4 and 5.1.5, while its built-in concurrency directly enabled concurrent batch prompting.

We employed two complementary optimization strategies. First, concurrent batch prompting accelerates LLM inference by processing multiple queries simultaneously

[12] https://github.com/ollama/ollama/tree/v0.12.3
[13] https://github.com/langchain-ai/langchain/tree/langchain-ollama==0.3.10

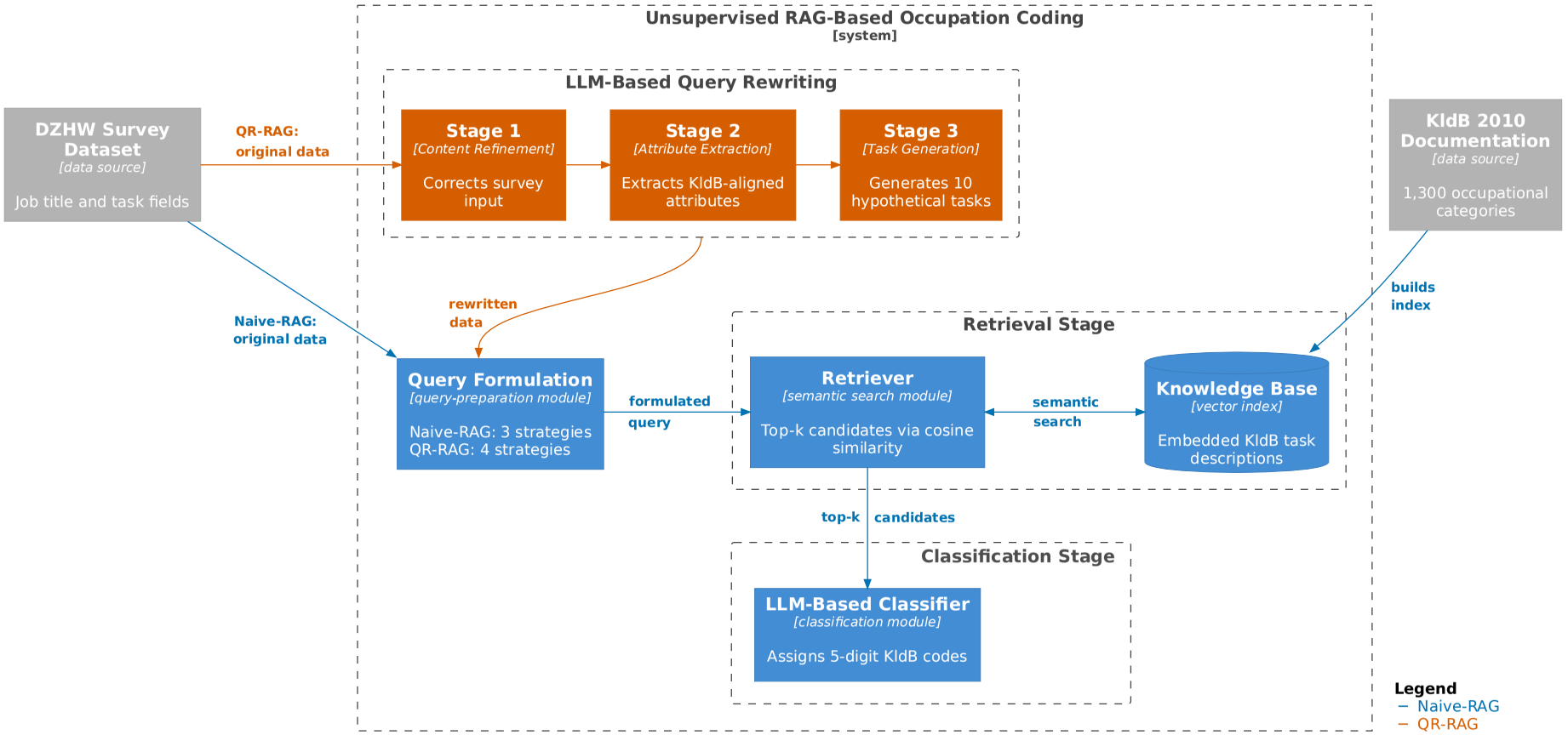


Figure 4.1: Pipeline-Design Overview: Naive-RAG and QR-RAG. Both approaches share three stages—query formulation, retrieval, and classification—and query the same knowledge base. QR-RAG prepends an LLM-based query-rewriting phase that transforms original survey data through three sequential inner stages before query formulation; Naive-RAG bypasses this phase entirely. Query formulation offers three strategies in Naive-RAG and four in QR-RAG.

across inference stages, with Ollama configuration parameters (Table A.3) tuned individually for QR-LLMs (Table 5.4) to balance throughput and memory consumption. Second, quantization reduces memory requirements through (i) 4-bit weight quantization for QR-LLMs, achieving over 50% VRAM reduction, and (ii) 8-bit key-value cache quantization with Flash Attention for all inference stages, minimizing quality degradation. Because the optimal QR-LLM serves as the LLM-based classifier for both approaches, both the 4-bit quantization and concurrent batch prompting apply to QR-RAG's query-rewriting stage and the shared classification stage. Together, these optimizations enabled the comprehensive experimental evaluation within local resource constraints.

4.2 Knowledge-Base Construction

With the design decisions established, we now detail the retrieval corpus underlying both RAG approaches. Both approaches query a shared knowledge base constructed from the official KldB 2010 documentation in its 2020 revision. Section 1.1.1 introduced the KldB structure and hierarchical organization. This section documents the source-material acquisition, field-extraction procedures, knowledge-base document construction, and vector-database indexing that produced the retrieval corpus.

4.2.1 Source Materials

Knowledge-base construction begins with acquiring the official KldB documentation. We obtained the complete set of 5-digit codes from the DKZ-Downloadportal[14] provided by the German Federal Employment Agency (BA). Specifically, we extracted these codes from the file "Gesamtberufsliste der Bundesagentur für Arbeit.xlsx"[15], representing all valid categories in the 2020 revision of the KldB 2010. Using this code list, we downloaded the corresponding authoritative textual descriptions—published as individual portable document format (PDF) files[16]—yielding 1,300 source documents. Each source document follows a standardized six-section structure, though document length varies (1–2 pages depending on occupational complexity). Figure 4.2 illustrates this structure using code "43414" ("Berufe in der Softwareentwicklung – hoch komplexe Tätigkeiten"—"Occupations in Software Development – Highly Complex Activities") as a representative example. The six sections comprise:

1. **Occupational-Area Header:** Broad 1-digit category designation ("Berufsbereich"). The example shows "Berufsbereich 4: Naturwissenschaft, Geografie und Informatik"—"Occupational Area 4: Natural Sciences, Geography and Computer Science".
2. **Code and Category Title:** 5-digit code combined with the full occupation designation, which includes the occupational type and complexity level. The example shows "Berufe in der Softwareentwicklung – hoch komplexe Tätigkeiten" (case-normalized).

[14] https://www.arbeitsagentur.de/institutionen/dkz-downloadportal

[15] Downloadportal der Bundesagentur für Arbeit, https://www.arbeitsagentur.de – Stand 04/2025.

[16] Documents follow the URL pattern: https://www.pub.arbeitsagentur.de/hst/services/statistik/000200/html/kldb-online/downloads/<5-digit-code>.pdf

3. **Content Summary:** Brief summary description ("Inhalt") defining the scope, boundaries, and characteristic requirements of the occupational category. The example contains two sentences.
4. **Task Descriptions:** Bulleted list ("Aufgaben, Tätigkeiten, Kenntnisse und Fertigkeiten, üblicherweise") detailing typical work activities, responsibilities, required domain knowledge, and technical competencies, typically containing seven bulleted items (10 items in the example). Despite the comprehensive German label, we refer to this section as task descriptions throughout this thesis, as it constitutes the primary substantive content of each document.
5. **Assigned Job Titles:** List ("Zugeordnete Berufe") of specific job titles with subcodes officially designated as Digital Classification Center (DKZ) job titles, providing representative (not exhaustive) examples of occupations classified under this code. The example lists four titles (e.g., "43414-124 Software-Architect").
6. **Excluded Job Titles:** List ("Nicht einzubeziehende Berufe") of DKZ job titles explicitly excluded from this category to clarify boundaries with similar occupational categories. This section may be absent if no common classification ambiguities exist. The example lists one excluded occupation ("43104-121 Informatiker/in"—"Computer Scientist").

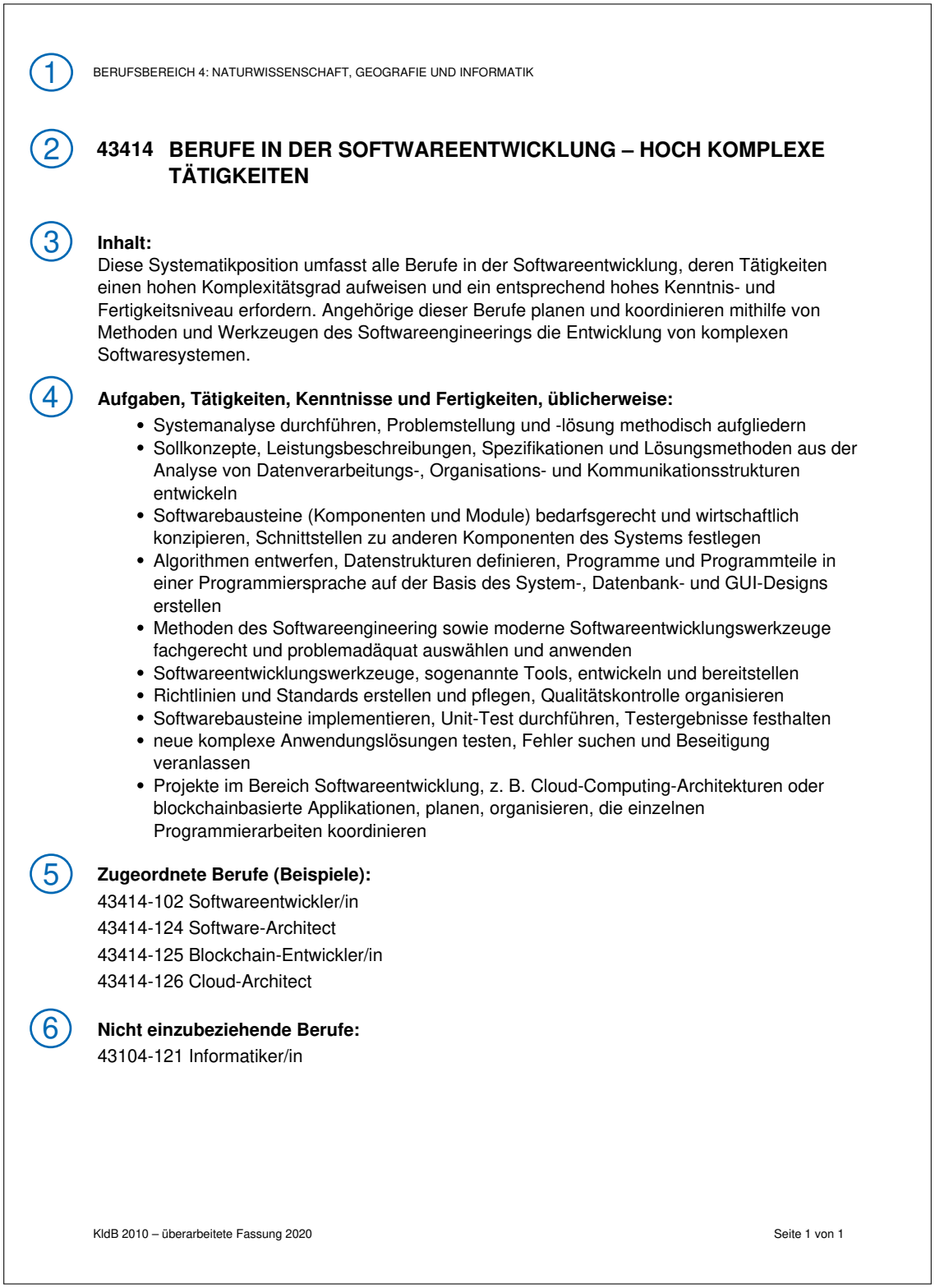
4.2.2 Field Extraction and Document Construction

The multi-section document structure presented a critical design choice: which content fields to extract for knowledge-base construction. While each source document contains six sections, not all provide equally useful retrieval signals for occupation coding. This section justifies extracting task descriptions as the sole content field, documents the extraction procedure, and describes the standardized document structure.

4.2.2.1 Field-Selection Rationale

Having identified the six-section structure of source documents, we now justify our decision to extract only task descriptions. We prioritized broad, category-level task descriptions as the sole content field for three interconnected reasons. First, task descriptions capture the activities, responsibilities, and competencies that distinguish one occupational category from another—likely providing the strongest semantic discriminator among available document sections for both retrieval and classification. Though aggregated at the category level rather than tailored to individual positions, we expected this content would provide sufficient discriminative power for survey-based occupation coding.

Second, formal task descriptions align with the predominant content type respondents typically provide in survey responses. This task-oriented focus in survey data—though expressed in colloquial language—was expected to create content overlap that facilitates retrieval matching. This alignment is reinforced by the query-rewriting approach: QR-RAG was designed to expand brief survey responses into hypothetical task descriptions—a strategy we expected to yield stronger query–document alignment when knowledge-base documents are similarly organized around task-oriented content.



1 BERUFSBEREICH 4: NATURWISSENSCHAFT, GEOGRAFIE UND INFORMATIK

2 **43414 BERUFE IN DER SOFTWAREENTWICKLUNG – HOCH KOMPLEXE TÄTIGKEITEN**

3 **Inhalt:**
Diese Systematikposition umfasst alle Berufe in der Softwareentwicklung, deren Tätigkeiten einen hohen Komplexitätsgrad aufweisen und ein entsprechend hohes Kenntnis- und Fertigkeitsniveau erfordern. Angehörige dieser Berufe planen und koordinieren mithilfe von Methoden und Werkzeugen des Softwareengineerings die Entwicklung von komplexen Softwaresystemen.

4 **Aufgaben, Tätigkeiten, Kenntnisse und Fertigkeiten, üblicherweise:**

- Systemanalyse durchführen, Problemstellung und -lösung methodisch aufgliedern
- Sollkonzepte, Leistungsbeschreibungen, Spezifikationen und Lösungsmethoden aus der Analyse von Datenverarbeitungs-, Organisations- und Kommunikationsstrukturen entwickeln
- Softwarebausteine (Komponenten und Module) bedarfsgerecht und wirtschaftlich konzipieren, Schnittstellen zu anderen Komponenten des Systems festlegen
- Algorithmen entwerfen, Datenstrukturen definieren, Programme und Programmteile in einer Programmiersprache auf der Basis des System-, Datenbank- und GUI-Designs erstellen
- Methoden des Softwareengineering sowie moderne Softwareentwicklungswerkzeuge fachgerecht und problemadäquat auswählen und anwenden
- Softwareentwicklungswerkzeuge, sogenannte Tools, entwickeln und bereitstellen
- Richtlinien und Standards erstellen und pflegen, Qualitätskontrolle organisieren
- Softwarebausteine implementieren, Unit-Test durchführen, Testergebnisse festhalten
- neue komplexe Anwendungslösungen testen, Fehler suchen und Beseitigung veranlassen
- Projekte im Bereich Softwareentwicklung, z. B. Cloud-Computing-Architekturen oder blockchainbasierte Applikationen, planen, organisieren, die einzelnen Programmierarbeiten koordinieren

5 **Zugeordnete Berufe (Beispiele):**
43414-102 Softwareentwickler/in
43414-124 Software-Architect
43414-125 Blockchain-Entwickler/in
43414-126 Cloud-Architect

6 **Nicht einzubeziehende Berufe:**
43104-121 Informatiker/in

KldB 2010 – überarbeitete Fassung 2020 Seite 1 von 1

Figure 4.2: Standardized Structure of KldB 2010 Documents. All 1,300 occupational descriptions follow the identical six-section template shown here. Blue ellipses with numeric annotations (1–6, ordered top to bottom) mark each section. Section 4 provides the task descriptions extracted for knowledge-base construction. Source: German Federal Employment Agency, https://www.pub.arbeitsagentur.de/hst/services/statistik/000200/html/kldb-online/downloads/43414.pdf.

Third, task descriptions offer practical implementation advantages. The bulleted task lists provide adequate textual content (averaging seven items per document) to support meaningful dense embeddings, with each list forming a coherent thematic unit focused on a single occupational category. Extracting only task descriptions rather than combining multiple sections preserves these coherent units and ensures each document fits within typical embedding-model context windows (512 tokens). Including additional content would have required document chunking—fragmenting the coherent categorical descriptions and complicating the indexing pipeline beyond this thesis's scope.

These advantages for task descriptions became clearer when we assessed the limitations of other available sections. Sections 1 and 2 (occupational-area headers and category titles) contained content highly similar across closely related occupational categories. The occupational-area header (e.g., "Berufsbereich 4: Naturwissenschaft, Geografie und Informatik") is shared by all sibling codes within the same 1-digit category, while category titles often share parent-level designations with related codes (e.g., all software-development occupations share "Berufe in der Softwareentwicklung"). We hypothesized this overlapping hierarchical information might decrease discrimination capability. Section 3 (content summaries) offered mixed descriptive content partially redundant with task descriptions, potentially reducing signal clarity. Sections 5 and 6 (assigned and excluded DKZ job titles) provided only exemplary sets, not exhaustive lists. We hypothesized these might dilute retrieval signals: assigned titles could introduce surface-form variations that match queries spuriously, while excluded titles—representing occupations explicitly not belonging to the category—could contaminate retrieval signals by embedding semantically similar but categorically incorrect content.

This field selection prioritized task-based semantic matching while avoiding content we hypothesized might introduce noise or ambiguity. Task descriptions provide more detailed and operationally relevant occupational characterizations than other document sections, which primarily offer structural metadata (codes, hierarchical labels) or boundary clarifications (assigned and excluded DKZ job titles) for taxonomic organization rather than semantic retrieval. While boundary-clarification information could theoretically enhance classification accuracy by providing positive and negative exemplars within LLM classification prompts, we did not extract or use this data, as integrating it fell outside this thesis's scope.

4.2.2.2 Task-Description Extraction

With task descriptions identified as the target content field, we implemented the technical extraction process. We extracted task descriptions from all 1,300 KldB documents using `Docling`[17] [Auer et al., 2024]. We selected this parser after evaluating multiple Python PDF text extraction tools (including `pypdf` and `pdfplumber`). Docling better preserved the list-level structure of task descriptions than alternatives, retaining the bulleted formatting we hypothesized would aid retrieval and classification, though some manual correction remained necessary. Alternative parsers concatenated list items into unstructured paragraphs without separators or markup,

[17] https://github.com/docling-project/docling/tree/v2.57.0

making discrete task boundaries unrecoverable and potentially obscuring the structured information present in source documents.

We applied manual post-processing to correct residual parsing errors. Manual inspection revealed occasional concatenation issues where Docling merged separate list items. Though we did not systematically quantify error frequency, we corrected all identified instances to ensure each original list item was accurately represented. This verification process ensured high fidelity to the official taxonomy content and structure.

4.2.2.3 Document Structure

After extracting task descriptions from source PDFs, we structured them into standardized knowledge-base documents. We formatted each extracted task-description list as a plain-text document following a standardized template. Listing 4.1 shows the general structure, while Listing 4.2 illustrates actual content from code "43414", introduced in Figure 4.2.

Listing 4.1: Template for Knowledge-Base Documents. All 1,300 documents follow this structure: a semantic header followed by bulleted task items. The placeholder n represents variable list length across occupational categories.

```
Berufsaufgaben:
- {Task 1}
- {Task 2}
- ...
- {Task n}
```

The header "Berufsaufgaben:" ("Occupational Tasks") signals task-oriented content, followed by bulleted items preserving the original list structure from source PDFs. Each item corresponds to one extracted list entry, stored as a single line terminated by a newline character ("\n"). We applied no additional preprocessing (e.g., normalization, stemming, or stopword removal), maintaining fidelity to the authoritative source content. This extraction and structuring yielded 1,300 task-oriented knowledge-base documents—one per 5-digit KldB code—ready for vector-database indexing.

4.2.3 Vector-Database Implementation

The extracted knowledge-base documents required indexing in a vector database to enable semantic similarity search during retrieval. This subsection documents the vector-store selection, embedding-model configuration, and document indexing strategy that produced the retrieval infrastructure.

4.2.3.1 Vector-Store Selection

The vector-database implementation required selecting an appropriate storage backend capable of efficient semantic search at the scale of our knowledge base. We employed `Chroma`[18] as the vector-database backend. We selected this open-source vector

[18] https://github.com/chroma-core/chroma/tree/1.1.1

Listing 4.2: Knowledge-Base Document for Code 43414. Extracted content for "Berufe in der Softwareentwicklung – hoch komplexe Tätigkeiten" (10 task items), corresponding to Figure 4.2. Line breaks within items result from display constraints; each bulleted item terminates with a newline character in storage.

```
Berufsaufgaben:
- Systemanalyse durchführen, Problemstellung und -lösung methodisch
  aufgliedern
- Sollkonzepte, Leistungsbeschreibungen, Spezifikationen und Lösungsmethoden
  aus der Analyse von Datenverarbeitungs-, Organisations- und
  Kommunikationsstrukturen entwickeln
- Softwarebausteine (Komponenten und Module) bedarfsgerecht und
  wirtschaftlich konzipieren, Schnittstellen zu anderen Komponenten des
  Systems festlegen
- Algorithmen entwerfen, Datenstrukturen definieren, Programme und
  Programmteile in einer Programmiersprache auf der Basis des System-,
  Datenbank- und GUI-Designs erstellen
- Methoden des Softwareengineering sowie moderne
  Softwareentwicklungswerkzeuge fachgerecht und problemadäquat auswählen und
  anwenden
- Softwareentwicklungswerkzeuge, sogenannte Tools, entwickeln und
  bereitstellen
- Richtlinien und Standards erstellen und pflegen, Qualitätskontrolle
  organisieren
- Softwarebausteine implementieren, Unit-Test durchführen, Testergebnisse
  festhalten
- neue komplexe Anwendungslösungen testen, Fehler suchen und Beseitigung
  veranlassen
- Projekte im Bereich Softwareentwicklung, z. B.
  Cloud-Computing-Architekturen oder blockchainbasierte Applikationen,
  planen, organisieren, die einzelnen Programmierarbeiten koordinieren
```

store for its straightforward integration with the `LangChain`[19] framework (`langchain-chroma`[20]), support for persistent storage, and efficient approximate nearest-neighbor search capabilities suitable for knowledge bases of this scale. Chroma implements HNSW graph-based indexing for fast retrieval and automatically normalizes embedding vectors to unit length, enabling cosine similarity as the metric for all semantic search operations. We used these configurations throughout all experiments.

4.2.3.2 Embedding-Model Configuration

With the vector store established, we configured the embedding models that transform documents into semantic vectors. We embedded knowledge-base documents using dual-encoder EMs that enable efficient retrieval by allowing pre-computation of document embeddings and fast semantic similarity matching at query time. We evaluated nine multilingual EMs selected for their German-language capabilities and diverse architectural characteristics, with complete specifications and selection

[19] https://github.com/langchain-ai/langchain-community/tree/libs/community/v0.3.31
[20] https://github.com/langchain-ai/langchain/tree/langchain-chroma==0.2.6

rationale documented in Section 5.1.4. Because dual-encoder models produce fixed-length vectors in model-specific semantic spaces, embeddings from different models are incompatible. We therefore maintained nine separate vector-database instances—one per EM—ensuring consistency between knowledge-base embeddings and query embeddings generated by the same model during retrieval.

Several evaluated models support instruction-finetuned embedding with task-specific prefixes that guide the embedding process toward particular use cases such as retrieval, classification, or clustering. For models with this capability, we applied their default retrieval-task prefixes to both queries and documents during embedding, following manufacturer specifications detailed in Table 5.2. Listing 4.3 illustrates how document prefixes were prepended to the knowledge-base document template for two instruction-aware models.

Listing 4.3: Knowledge-Base Documents with Instruction Prefixes. Task-specific document prefixes prepended to the knowledge-base document template for instruction-aware EMs. The blue prefix ("passage:") indicates Multilingual-E5-Large; the orange prefix ("title: none | text:") indicates EmbeddingGemma. Each prefix is followed by whitespace, then the semantic header and task list. Models without instruction-finetuning capabilities embed documents without prefixes.

```
# Multilingual-E5-Large
passage: Berufsaufgaben:
- {Task 1}
- {Task 2}
- ...
- {Task n}

# EmbeddingGemma
title: none | text: Berufsaufgaben:
- {Task 1}
- {Task 2}
- ...
- {Task n}
```

We configured all models to use their maximum supported context windows, ranging from 512 tokens for several models to 40,960 tokens for Qwen3-Embedding (Table 5.2). All embedding operations used deterministic settings, with complete execution parameters documented in Appendix A.2.2.

4.2.3.3 Document Indexing Strategy

The final implementation decision concerned document granularity: whether to index complete documents or split them into smaller chunks. We indexed documents without chunking, treating each complete task-description list as a single retrievable unit. During indexing, we attached each document's corresponding 5-digit code as metadata, enabling its use in LLM classification prompts.

This no-chunking design decision was intended to satisfy both technical constraints and evaluation requirements. Before indexing, we verified document compatibility

with embedding-model context windows. Using `TikToken`[21] with the standard GPT-2 tokenizer as a conservative proxy—GPT-2 tends to overcount tokens for German text relative to multilingual tokenizers—we found that nearly all 1,300 documents remained under 512 tokens, the minimum context window among evaluated models. Ollama raises exceptions when embedding inputs exceed a model's context window; the absence of any such exceptions confirmed that all documents fit within their respective context windows without truncation. Document-level indexing therefore avoided fragmentation while preserving the coherent thematic units established during document construction (Section 4.2.2.3). Additionally, the evaluation design required retrieving complete occupation descriptions as candidate matches for classification, making document-level indexing the natural granularity and simplifying implementation compared to chunked retrieval with metadata-based reconstruction.

We embedded all knowledge-base documents once at system initialization and persisted the resulting vector-database instances to disk. This one-time embedding strategy enabled reuse across all experimental runs without re-embedding, ensuring consistency and reliability while avoiding redundant computational overhead during evaluation.

4.3 Shared Pipeline Components

Both RAG approaches share identical retrieval and classification implementations, with query preparation constituting the sole experimental variable separating them. Naive-RAG formulates queries directly from original survey data, while QR-RAG formulates them from LLM-based rewritten data. The two subsections that follow detail these shared components: the retrieval stage, which queries the knowledge base and surfaces ranked candidates (Section 4.3.1), and the classification stage, which assigns the final KldB code from those candidates (Section 4.3.2).

4.3.1 Retrieval Stage

The retrieval stage receives a formulated query string and returns the top-k ranked candidates from the knowledge base, each accompanied by its task-oriented document and 5-digit KldB code. The incoming query string is first transformed into a dense vector representation using the same dual-encoder EM instance that produced the corresponding vector store (Section 4.2.3.2), ensuring that query and document vectors reside in the same semantic space. For instruction-aware models, a task-specific query prefix is prepended to the query string immediately before embedding, guiding the model toward retrieval-oriented representations; for example, Multilingual-E5-Large applies "query:" while EmbeddingGemma applies "task: search result | query:" as their respective query prefixes. Table 5.2 documents the complete prefix configuration for all nine evaluated models.

The embedded query vector is matched against pre-embedded task descriptions in the model-specific Chroma instance using cosine similarity over HNSW-indexed vectors (Section 4.2.3.1), returning the top-k highest-scoring candidates as document–score pairs. We treated the retrieval depth k as an experimental variable; Section 5.2.3

[21] https://github.com/openai/tiktoken/tree/0.12.0

establishes $k = 20$ as the optimal retrieval depth, confirmed by parallel findings in Section 5.3, and applied uniformly in all system-level experiments.

Each retrieved candidate exposes two components for downstream use: the task-oriented document as plain text (Section 4.2.2.3) and the 5-digit KldB code attached as document metadata during indexing (Section 4.2.3.3). These components serve distinct roles in the classification stage: task descriptions provide semantic context for LLM-based candidate comparison, while the KldB code enables direct prediction of the classification label without additional lookup. The retrieval stage passes all k candidates to the classification stage regardless of individual similarity scores, delegating final candidate selection to the LLM-based classifier.

4.3.2 Classification Stage

The classification stage receives the top-k retrieved candidates alongside the occupational data for the instance being classified and produces a single predicted 5-digit KldB code. The LLM-based classifier employed here is the optimal QR-LLM identified in Section 5.3, serving a dual role in our system: as query rewriter within QR-RAG's inner pipeline (Section 4.5.1) and as the shared classifier for both approaches. We expected its capacity to handle occupational data during query rewriting to transfer directly to classification, which requires analogous semantic reasoning to identify the best-matching code among the retrieved candidates.

Listing 4.4 presents the complete classification prompt template. This prompt is formulated entirely in German, ensuring linguistic consistency with its injected content. Its two-block structure instantiates the RAG paradigm directly: the first block presents the formatted candidates block—each entry pairing a KldB code with its knowledge-base task descriptions—as evidence from semantic search; the second block presents the job title and task content of the instance being classified as the subject for code assignment. The output constraint requesting only the 5-digit code without explanation reduces inference latency and enables deterministic output parsing by extracting the first 5-digit numeric sequence from the model response. Each instantiated prompt fits comfortably within the context window configured for the classifier (adapted to 19k tokens, as documented in Section 5.5), ensuring that no candidate is truncated at the prompt level. The extracted code constitutes the final classification output for each instance, evaluated against ground-truth labels using the metrics defined in Section 5.1.3.

Before prompt construction, each retrieved candidate is formatted as a labeled text block following Listing 4.5, pairing the KldB code with its associated task descriptions to form a single structured candidates block for injection into the *formatted_candidates* placeholder from Listing 4.4. We apply a 5,000-character safety threshold to each candidate's document prior to formatting as a precautionary measure; this guard was never triggered in practice, as all 1,300 task descriptions remain well within this limit. All k candidates are consequently injected in full for every classification instance.

The *occupational_data* placeholder from Listing 4.4 receives the job title and task content for the instance being classified, assembled independently of the query formulation used for retrieval. For Naive-RAG, this comprises the original job title and both original task-description fields; for QR-RAG, it comprises the Stage 2

Listing 4.4: Classification Prompt Template. The German-language prompt instructs the LLM-based classifier to identify the best-matching code from semantic search results. The *formatted_candidates* placeholder receives the formatted candidates block (Listing 4.5), presenting KldB codes alongside their knowledge-base task descriptions; the *occupational_data* placeholder receives job title and task content for the instance being classified (Listing 4.6). The explicit output constraint requests only the 5-digit code without explanation, minimizing response length and enabling deterministic parsing.

```
Du bist ein Assistent für die Klassifikation deutscher Berufe.

Deine Aufgabe ist es, aus den Ergebnissen einer Semantischen Suche,
die richtige 5-stellige 'kldb_id' zu den gegebenen Berufsdaten
zu finden.

Gib NUR die 'kldb_id' zurück, die am besten zu den Berufsdaten
passt. KEINE Erläuterungen und KEINE Überschrift, NUR die 'kldb_id'!

Ergebnisse der Semantischen Suche:
-----
{formatted_candidates}      # formatted list of k retrieved KldB candidates
-----

Berufsdaten (Berufsbezeichnung, beruflicher Tätigkeitsbereich
und berufliche Aufgaben):
{occupational_data}         # job title and task content (fixed structure)

Klassifikation 'kldb_id':
```

Listing 4.5: Candidate Formatting Template for Classification. Each retrieved candidate is rendered in this structure before injection into the *formatted_candidates* placeholder in Listing 4.4. The 5-digit code sourced from document metadata and the task descriptions from the knowledge base are presented together, enabling the classifier to reference both the candidate code and its associated occupational content.

```
Inhalt von kldb_id={kldb_id}:   # 5-digit KldB code (from metadata)
{doc_text}                      # task descriptions (from knowledge base)
```

transformed title and all 10 hypothetical task sentences. Stage 2 occupational attributes used in Schema-Data query formulations—such as industry, specialization, and work setting—are excluded from the classification input; the classifier therefore consistently operates over the same categories of occupational information for both approaches. Listing 4.6 illustrates this structure for both approaches, showing how the injected content differs despite an identical prompt template. Listing 4.7 instantiates the complete classification prompt using the QR-RAG approach and

the continuous running example from Listing 4.10, rendering the prompt's two-block structure concrete.

Listing 4.6: Classification Query Injection Across Both Approaches. Job title and task content injected into the *occupational_data* placeholder from Listing 4.4 for both RAG approaches, applied uniformly regardless of which query formulation was used for retrieval. In Naive-RAG (blue), the original job title is followed by both original survey task fields as bulleted items. In QR-RAG (orange), the Stage 2 transformed title is followed by 10 hypothetical task sentences.

```
# Naive-RAG
{job_title_original}:
- {tasks_1_original}
- {tasks_2_original}

# QR-RAG
{job_title_transformed}:  # from QR Stage 2
- {hypothetical_task_1}   # from QR Stage 3
- {hypothetical_task_2}
- ...
- {hypothetical_task_10}
```

4.4 Naive-RAG Implementation

Naive-RAG implements a direct three-stage pipeline without intermediate query transformation. Original survey data enters the query formulation stage, which composes a retrieval query string from available input fields using one of three predefined strategies. The formulated query then passes directly to the shared retrieval stage (Section 4.3.1) and subsequently to the shared classification stage (Section 4.3.2). This design tests whether participant language in its original form—often informal, abbreviated, or incomplete—suffices for accurate occupation coding without LLM-based query rewriting.

Query formulation is the sole architecture-specific component of the Naive-RAG pipeline. Three strategies are implemented, each composing a retrieval query string from a different subset of available survey-input fields: Title-Only, Tasks-Only, and Title+Tasks. These strategies operationalize the context-density dimension of information richness (Section 1.3.3), ranging from minimal-context queries using the job title alone to full-context queries incorporating the title and both task-description fields. All three formulations share a common German-language search directive, "Suche Berufsaufgaben für:" ("Search occupational tasks for:"), which orients embedding toward task-oriented retrieval against the knowledge base, consistent with the retrieval query structure described in Section 4.3.1. Their evaluation rationale and role within the experimental design are detailed in Section 5.1.6; this section documents their string-level implementation. Listing 4.8 presents the structural templates for all three formulations.

Listing 4.7: Classification Prompt Example. The QR-RAG classification prompt instantiated using the continuous running example from Listing 4.10. Two of k retrieved candidates are shown with representative task descriptions; the remaining candidates are omitted for brevity. The *occupational_data* block from Listing 4.4 uses the Stage 2 transformed title and two of the 10 hypothetical task sentences.

```
Du bist ein Assistent für die Klassifikation deutscher Berufe.

Deine Aufgabe ist es, aus den Ergebnissen einer Semantischen Suche,
die richtige 5-stellige 'kldb_id' zu den gegebenen Berufsdaten
zu finden.

Gib NUR die 'kldb_id' zurück, die am besten zu den Berufsdaten
passt. KEINE Erläuterungen und KEINE Überschrift, NUR die 'kldb_id'!

Ergebnisse der Semantischen Suche:
-----
Inhalt von kldb_id=81763:              # candidate 1 of k
Berufsaufgaben:
- Ernährungskonzepte bei bestimmten Krankheiten (z. B. Allergien,
  Rheuma, Gicht, Hautkrankheiten, Gefäßkrankheiten, Bluthochdruck,
  Darmerkrankungen) aufstellen und im Rahmen von Diät- und
  Ernährungstherapien umsetzen
- Anamnesen, individuelle Nährstoffanalysen und Analysen von
  Mangelzuständen durchführen, Diagnose stellen
- ...

Inhalt von kldb_id=81783:              # candidate 2 of k
Berufsaufgaben:
- sportmedizinische Tests durchführen, z. B. Funktionsstörungen an
  Muskeln und Gelenken mittels isokinetischer Geräte erkennen
- Übungen individuell auf Patienten abstimmen, Behandlungs- und
  Ernährungspläne oder spezielle Diätpläne für Diabetiker erstellen

...                                    # candidates 3--k omitted for brevity

-----

Berufsdaten (Berufsbezeichnung, beruflicher Tätigkeitsbereich
und berufliche Aufgaben):
Ernährungstherapeut:
- Ernährungspläne für Patienten mit Essstörungen im stationären Bereich
  individuell erstellen und anpassen
- Ernährungsberatung in Einzel- und Gruppengesprächen durchführen und
  dokumentieren
- ...

Klassifikation 'kldb_id':
```

Listing 4.8: Naive-RAG Query Formulation Templates. Structural templates for all three Naive-RAG query formulations. Both task-description fields are included unconditionally across Tasks-Only and Title+Tasks—when the second field is absent, its bullet item is empty.

```
# 1. Title-Only
Suche Berufsaufgaben für: {job_title_original}

# 2. Tasks-Only
Suche Berufsaufgaben für:
---
- {tasks_1_original}
- {tasks_2_original}
---

# 3. Title+Tasks
Suche Berufsaufgaben für:
---
{job_title_original}:
- {tasks_1_original}
- {tasks_2_original}
---
```

The *Title-Only* strategy constructs a single-line query from the original job title field alone. The *Tasks-Only* strategy constructs a block query from both original task-description fields, omitting the job title; both fields are included unconditionally regardless of content, so instances where the second task field is absent contribute an empty bullet item rather than being filtered out. The *Title+Tasks* strategy combines the original job title with both task-description fields into a full-context block query (task fields included unconditionally, as in Tasks-Only), providing the maximum available context from original survey data.

4.5 QR-RAG Implementation

QR-RAG extends the direct pipeline of Naive-RAG by introducing a query-rewriting phase designed to bridge the register dimension of the semantic gap (Section 1.1.2). The theoretical foundation for this phase is query–document alignment [Lee, 2025]: to effectively retrieve KldB job tasks from the knowledge base, the query must match the target documents in both semantic meaning and structural format. To achieve this alignment, the optimal QR-LLM drives the entire inner pipeline, first cleaning and normalizing the survey input, then generating semantic enrichments such as structured occupational fields and hypothetical job tasks.

Rather than executing these transformations through a single monolithic LLM prompt, we implemented a three-stage architecture in which each stage employs a specialized prompt: content refinement (Section 4.5.1.1), occupational attribute extraction (Section 4.5.1.2), and hypothetical task generation (Section 4.5.1.3). This design decision addresses the inherent limitations of LLMs. Because we process multiple survey instances simultaneously via batch prompting, combining all instructions

into one prompt would result in excessive token lengths and high task complexity, ultimately degrading generation quality. By dividing the workload, the system was designed to maintain higher-quality outputs than a single monolithic prompt would produce. To make this pipeline computationally viable on local hardware, we orchestrate these stages using a concurrent batch-prompting architecture (Section 4.5.2). Once the survey input is fully transformed into a formal representation, it feeds into one of four query-formulation strategies (Section 4.5.3) before entering the shared retrieval and classification stages (Sections 4.3.1 and 4.3.2).

4.5.1 Query-Rewriting Pipeline

The query-rewriting pipeline executes its three stages sequentially. During each stage, the system submits a single prompt containing a batch of survey instances to the optimal QR-LLM, then forwards the parsed JSON response to the next stage. Across the entire pipeline, the batch size remains fixed at five instances per prompt. We initially tested larger batches but retained the smaller size to prevent context-window overflows and to improve the alignment of batched input–output pairs. This decision aligns with Cheng et al. [2023], who identified batch sizes of five to six as the optimal trade-off between token throughput and task performance. To maximize efficiency, prompt requests are dispatched concurrently (Section 4.5.2), though per-model limits are enforced to respect available VRAM capacity (Table 5.4). The full German-language prompt templates—along with their iterative development process, few-shot sourcing, and structural anatomy—are provided in Appendix A.3. In the following, we detail the transformation logic and input–output schemas for each stage, tracking a single record through the pipeline as a continuous running example.

4.5.1.1 QR Stage 1: Content Refinement

Stage 1 performs content refinement, correcting surface-level defects in the original survey input before any structural transformation. It processes batches of survey records—each containing a job title, two task-description fields, and an additional integer identifier—and returns a corrected record per instance. Six correction operations are applied in the following order: (1) encoding repair, replacing corrupted characters with the corresponding German umlauts or ß where unambiguous; (2) preservation of content delimiters, retaining commas and semicolons used to enumerate distinct occupational content; (3) case normalization according to German orthographic conventions; (4) gender-form standardization, converting all occupational designations to masculine form to align with KldB conventions; (5) abbreviation expansion, resolving both general and domain-specific abbreviations where context permits; and (6) spelling and grammar correction without altering content or structure. Operations (1), (3), and (5) each target a documented LLM-addressable issue: character-encoding degradation (Section 3.3.2.5), inconsistent capitalization patterns (Section 3.3.2.4), and unknown abbreviations (Section 3.3.2.3).

The prompt in Listing A.1 instructs the QR-LLM to apply these operations in the stated order so that earlier corrections inform later ones, while ten annotated few-shot input–output pairs explicitly guide the model's refinement behavior. As formalized in the input–output schemas (Listing 4.9), each instance yields a structured record that preserves the job title and both task descriptions—either corrected or

returned unmodified—alongside the integer identifier. This identifier maintains strict record alignment across batch-parallel instances because, as observed during prompt development, the QR-LLM does not consistently preserve input ordering within batch responses. Finally, Listing 4.10 introduces a running example that tracks a single record through all three stages to demonstrate the pipeline's effects.

Listing 4.9: QR Stage 1 Input–Output Schemas. Both schemas share an identical four-field structure. The input provides the survey data, while the output schema presents the corrected text. The integer identifier passes through both schemas unchanged to ensure record alignment is maintained across batch-parallel instances.

```
// Input Schema (per record)
{
  "id":        integer,  // identifier for input-output alignment
  "job_title": string,   // original job title
  "tasks_1":   string,   // original task description 1
  "tasks_2":   string    // original task description 2
}

// Output Schema (per record)
{
  "id":        integer,  // unchanged from input
  "job_title": string,   // corrected or unchanged
  "tasks_1":   string,   // corrected or unchanged
  "tasks_2":   string    // corrected or unchanged
}
```

4.5.1.2 QR Stage 2: Occupational Attribute Extraction

Stage 2 performs occupational attribute extraction, transforming the corrected Stage 1 record into a structured, KldB-aligned representation. It is the most complex stage, requiring the QR-LLM to handle four distinct transformation tasks. The model must: (i) identify the primary occupational role when multiple roles are present; (ii) relocate misplaced content between fields; (iii) extract contextual occupational attributes; and (iv) assign a requirement level and hierarchy role. Task (ii) targets misplaced task descriptions (Section 3.3.2.2); the remaining two LLM-addressable issues—expression variability and semantic duplicates—are not addressed at any pipeline stage.

The prompt in Listing A.2 instructs the QR-LLM to perform these complex transformations, while eight annotated few-shot input–output pairs explicitly guide the model's extraction behavior. To support task (iv), the prompt supplies inline KldB-2010 domain knowledge—specifically, the four requirement levels and three hierarchy roles with their defining signal terms. As formalized in the input–output schemas (Listing 4.11), each instance yields a structured, KldB-aligned record where the two task-description fields from Stage 1 are consolidated into a single *job_tasks* field. Extracted attributes default to empty strings when no explicit evidence is present. Crucially, the predicted *requirement_level* (mapped to values "1" to "4") and *hierarchy_role* (mapped to "Fachkraft"—skilled worker, "Aufsichtskraft"—supervisor,

Listing 4.10: QR Stage 1 Transformation Example. Serving as a continuous running example across all three pipeline stages, this record demonstrates the immediate impact of Stage 1 corrections. By applying encoding repair (1), corrupted character sequences (ï¿½) are resolved into valid German characters; simultaneously, gender-form standardization (4) converts the feminine suffix to the target masculine form.

```
// Input
{
  "id":        1,
  "job_title": "ErnÃ¯Â¿Â½hrungstherapeutin",
  "tasks_1":   "Beratung, LehrkÃ¯Â¿Â½che, Gruppen ... mit jungen
  ↪ essgestÃ¯Â¿Â½rten Patienten",
  "tasks_2":   "therapeutische TÃ¯Â¿Â½tigkeit im stationÃ¯Â¿Â½ren Setting"
}

// Output (operations 1 and 4 applied)
{
  "id":        1,
  "job_title": "Ernährungstherapeut",
  "tasks_1":   "Beratung, Lehrküche, Gruppen ... mit jungen essgestörten
  ↪ Patienten",
  "tasks_2":   "therapeutische Tätigkeit im stationären Setting"
}
```

or "Führungskraft"—manager) serve strictly as intermediate heuristics, not final classification targets. Instead, they ensure the complexity and seniority of the generated tasks in Stage 3 mirror the corresponding KldB categories. Furthermore, when a survey response conflates multiple occupational roles (Section 3.3.3.1), the QR-LLM examines every input field to select the primary role by task-signal density and requirement level. It then isolates secondary roles and their tasks into *multi_role_data*, excluding them from downstream retrieval queries and classification inputs. Finally, Listing 4.12 continues the running example to demonstrate these extractions.

4.5.1.3 QR Stage 3: Hypothetical Task Generation

Stage 3 translates the structured Stage 2 output into standardized task descriptions by generating exactly 10 distinct hypothetical job-task sentences per instance. This generation is conditioned on eight retained attributes: job title, job tasks, requirement level, hierarchy role, industry, specialization, work setting, and employer type. Because geographic locations do not correlate with KldB categories and secondary roles would introduce conflicting signals, location and multi-role data are deliberately excluded to prevent retrieval degradation. To bridge the vocabulary gap between informal survey language and formal classification standards—an alignment we expected to improve retrieval quality—the generated tasks are formulated as active infinitive constructions in the linguistic style of KldB task descriptions (e.g., "Unterlagen prüfen" rather than "Prüfung von Unterlagen"—"check documents" versus "document review").

Listing 4.11: QR Stage 2 Input–Output Schemas. Taking the four-field corrected output of Stage 1 as input, this schema constructs a detailed, KldB-aligned representation. Because not all records contain complete information, unsubstantiated attributes default to empty strings. Crucially, to accurately calibrate the complexity of downstream task generation in Stage 3, the schema also captures the predicted requirement-level and hierarchy-role fields.

```
// Input Schema (per record, from Stage 1)
{
  "id":        integer,  // identifier for input-output alignment
  "job_title": string,   // corrected job title
  "tasks_1":   string,   // corrected task description 1
  "tasks_2":   string    // corrected task description 2
}

// Output Schema (per record)
{
  "id":               integer, // unchanged from input
  "job_title":         string, // core title of primary role
  "job_tasks":         string, // tasks of primary role only
  "requirement_level": string, // "1" to "4" or empty
  "hierarchy_role":    string, // role category or empty
  "industry":          string, // sector or empty
  "specialization":    string, // domain specialization or empty
  "work_setting":      string, // physical work environment or empty
  "employer_type":     string, // employer category or empty
  "location":          string, // geographic location or empty
  "multi_role_data":   string  // secondary roles or empty
}
```

The prompt in Listing A.3 instructs the QR-LLM to perform this transformation, while two annotated few-shot input–output pairs guide the model's task-generation behavior. As formalized in the input–output schemas (Listing 4.13), each instance yields a three-field record comprising the integer identifier, the unchanged job title from Stage 2, and the newly generated task array. Finally, Listing 4.14 continues the running example introduced in Listing 4.10 to demonstrate the pipeline's completed effects.

4.5.2 Concurrent Batch-Prompting Architecture

Processing thousands of survey records autoregressively using high-parameter LLMs creates a severe computational bottleneck, particularly on consumer-grade hardware. To make local inference practically viable, the query-rewriting pipeline employs a concurrent batch-prompting architecture. This design maximizes token throughput and GPU saturation by bundling multiple survey instances into single prompts and dispatching them in parallel. As illustrated in Figure 4.3, this execution architecture comprises three primary components: a batch orchestrator for parallel task distribution, a batch fallback handler for automated error recovery, and a persistence manager.

Listing 4.12: QR Stage 2 Transformation Example. Continuing the running example from Stage 1, the pipeline consolidates the two original task-description fields into a single *job_tasks* field and extracts six distinct occupational attributes from the context. Because the input provides no explicit evidence for *employer_type*, *location*, or *multi_role_data*, these fields are safely returned empty.

```
// Input (Stage 1 output)
{
  "id":         1,
  "job_title": "Ernährungstherapeut",
  "tasks_1":    "Beratung, Lehrküche, Gruppen ... mit jungen essgestörten
  ↪ Patienten",
  "tasks_2":    "therapeutische Tätigkeit im stationären Setting"
}

// Output (Stage 2 extraction)
{
  "id":                1,
  "job_title":         "Ernährungstherapeut",
  "job_tasks":         "Beratung, Lehrküche, Gruppenarbeit mit jungen
  ↪ essgestörten Patienten, therapeutische Tätigkeit im stationären Setting",
  "requirement_level": "3",
  "hierarchy_role":    "Fachkraft",
  "industry":          "Gesundheitswesen",
  "specialization":    "Ernährungstherapie",
  "work_setting":      "Klinik, stationär",
  "employer_type":     "",
  "location":          "",
  "multi_role_data":   ""
}
```

The *batch orchestrator* serves as the primary coordination layer. It loads the original survey data and drives the sequential pipeline stages. For each stage, the orchestrator partitions the data into fixed 5-instance batches, constructs the appropriate LLM prompts, and dispatches them concurrently to the Ollama inference backend. To eliminate the need for isolated intermediate files, the orchestrator appends each stage's output directly into new columns of a single, shared pandas DataFrame, keeping all data transparent and accessible throughout the run.

Within each stage, data flows through a LangChain runnable sequence (Listing 4.15). Every stage employs the same 7-step execution pattern: (1) batch prompt construction, (2) LLM invocation, (3) message content extraction, (4) special-character cleaning, (5) reasoning trace removal, (6) JSON parsing with LLM-assisted error correction, and (7) identifier-alignment validation. The final validation step is critical to prevent data corruption: it verifies that the unique integer identifier assigned to each input record (Section 4.5.1) perfectly matches the identifier returned in the generated JSON object, ensuring strict input–output mapping. Because generation errors are inevitable, the pipeline wraps this entire sequence in two layers of recovery:

Listing 4.13: QR Stage 3 Input–Output Schemas. The input schema filters the eleven Stage 2 output fields down to nine, deliberately excluding location and multi-role data to prevent retrieval degradation. The output schema then defines a simplified three-field record designed to capture the array of generated task strings.

```
// Input Schema (per record, from Stage 2)
{
  "id":               integer, // identifier for input-output alignment
  "job_title":         string, // core title of primary role
  "job_tasks":         string, // tasks of primary role
  "requirement_level": string, // "1" to "4" or empty
  "hierarchy_role":    string, // role category or empty
  "industry":          string, // sector or empty
  "specialization":    string, // domain specialization or empty
  "work_setting":      string, // physical work environment or empty
  "employer_type":     string  // employer category or empty
}

// Output Schema (per record)
{
  "id":                    integer, // unchanged from input
  "job_title":              string, // unchanged from input
  "hypothetical_job_tasks": [
    string,                        // task 1
    ...                            // ...
    string                         // task 10
  ]
}
```

invocation-level retries catch transient connection drops, while sequence-level retries handle downstream parsing failures.

Despite these safeguards, batch processing introduces a specific vulnerability: a single JSON syntax error or context-window overflow can crash an entire 5-instance batch. When this occurs, the *batch fallback handler* automatically intervenes. It decomposes the failed batch into five individual queries and reprocesses them concurrently. This decomposition solves two problems simultaneously: it shrinks the prompt size to prevent context-window limits and drastically simplifies the generation task to mitigate complex syntax errors. Empirical testing during development indicated that this automated fallback recovered the vast majority of initially failed batches, substantially reducing the need for manual intervention.

Upon pipeline completion, the *persistence manager* serializes the final DataFrame to a Python pickle file. To guarantee experimental provenance, this file bundles the outputs with comprehensive metadata—including LLM configuration parameters, batch settings, execution statistics, and the exact prompt templates used. Crucially, this structured format allows us to systematically track experiments and directly compare results across different QR-LLM configurations.

Listing 4.14: QR Stage 3 Transformation Example. Conditioned on the filtered attributes from Stage 2, this final step generates exactly 10 distinct hypothetical task sentences. To bridge the vocabulary gap between informal survey language and formal taxonomy standards, the QR-LLM strictly formulates every generated task as an active infinitive construction.

```
// Input (Stage 2 output)
{
  "id":               1,
  "job_title":        "Ernährungstherapeut",
  "job_tasks":        "Beratung, Lehrküche, Gruppenarbeit mit jungen
  ↪ essgestörten Patienten, therapeutische Tätigkeit im stationären Setting",
  "requirement_level": "3",
  "hierarchy_role":   "Fachkraft",
  "industry":         "Gesundheitswesen",
  "specialization":   "Ernährungstherapie",
  "work_setting":     "Klinik, stationär",
  "employer_type":    ""
}

// Output (Stage 3 generation)
{
  "id":        1,
  "job_title": "Ernährungstherapeut",
  "hypothetical_job_tasks": [
    "Ernährungspläne für Patienten mit Essstörungen im stationären Bereich
  ↪ individuell erstellen und anpassen",
    "Ernährungsberatung in Einzel- und Gruppengesprächen durchführen und
  ↪ dokumentieren",
    "Ernährungssituationen von jungen Patienten mit Essstörungen analysieren
  ↪ und bewerten",
    "Therapeutische Maßnahmen im Rahmen der Ernährungsbehandlung planen und
  ↪ umsetzen",
    "Kooperation mit Ärzten, Psychologen und Pflegekräften zur ganzheitlichen
  ↪ Behandlung von Patienten herstellen",
    "Ernährungsverhalten und Essgewohnheiten von Patienten beobachten und
  ↪ dokumentieren",
    "Lehrküchenaktivitäten zur Verbesserung der Ernährungsbewusstheit und
  ↪ -kompetenz durchführen",
    "Ernährungsziele mit Patienten gemeinsam festlegen und deren Fortschritt ü
  ↪ berwachen",
    "Ernährungsberichte und Therapiepläne für die stationäre Behandlung
  ↪ erstellen und aktualisieren",
    "Fachliche Informationen zu Ernährung und Ernährungsstörungen an Kollegen
  ↪ und Fachkräfte weitergeben"
  ]
}
```

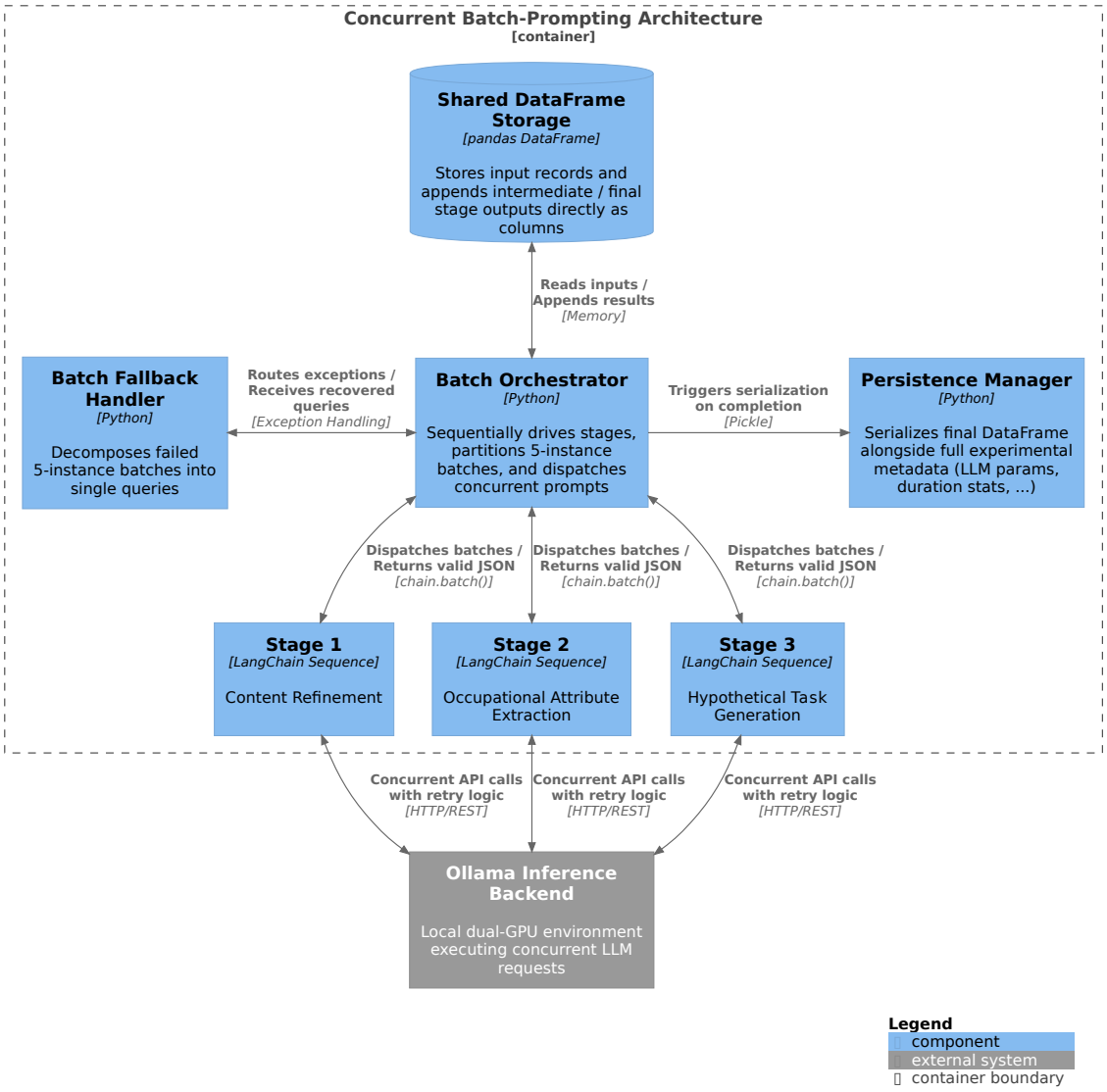


Figure 4.3: Concurrent Batch-Prompting Architecture. The batch orchestrator coordinates three sequential LangChain processing stages, dispatching 5-instance batches concurrently to the local Ollama inference backend. A batch fallback handler manages error recovery by decomposing failed batches into single queries, while intermediate and final outputs are appended to a shared pandas DataFrame and serialized by the persistence manager.

4.5.3 Query Formulations

QR-RAG implements four query-formulation strategies that compose retrieval query strings from the outputs of Stage 2 (Section 4.5.1.2) and Stage 3 (Section 4.5.1.3). They divide into two structural groups: two single-field formulations—Title-Only and Tasks-Only—each drawing on a single pipeline output, and two multi-field formulations—Title+Tasks and Schema-Data—each aggregating multiple outputs. Three of these strategies—Title-Only, Tasks-Only, and Title+Tasks—correspond structurally to their Naive-RAG counterparts (Section 4.4), differing only in that their content derives from LLM-rewritten data. Schema-Data is exclusive to QR-RAG as it

Listing 4.15: Shared 7-Step LangChain Runnable Sequence. While stage-specific behavior is injected via the prompt template, all three pipeline stages share this core 7-step LangChain architecture, shown here using Stage 1 as a representative example. To maximize fault tolerance, the sequence employs two distinct recovery layers: invocation-level retries handle transient LLM connection errors, while sequence-level retries manage downstream parsing or identifier-alignment failures.

```
chain_stage1 = (
    # 1. Batch prompt construction (stage-specific)
    prompt_stage1
    # 2. LLM invocation via Ollama
    | llm.with_retry(
        stop_after_attempt=100,
        retry_if_exception_type=(ResponseError,)
      )
    # 3. Message content extraction
    | StrOutputParser()
    # 4. Special character cleaning
    | ComprehensiveCleaningParser()
    # 5. Reasoning trace removal
    | (lambda x: x.split("</think>")[-1].strip()
                 if "</think>" in x else x)
    # 6. JSON parsing with LLM-assisted error correction (stage-specific)
    | fixing_parser_stage1
    # 7. ID-alignment validation
    | IdAlignmentValidator
).with_retry(stop_after_attempt=5)  # Sequence-level retry
```

requires the occupational attributes produced by Stage 2—fields that original survey data does not provide. All four formulations share the German-language search directive "Suche Berufsaufgaben für:" (Section 4.4), orienting the embedding toward task-oriented knowledge-base documents. Listing 4.16 presents the four structural templates side by side, making their data sources and structural patterns directly comparable; Listing 4.17 instantiates each template using the continuous running example from Listing 4.10, rendering the produced query strings concrete.

The two *single-field formulations* isolate specific occupational dimensions but introduce an information asymmetry: whereas Naive-RAG's single-field queries draw on each input field in isolation, QR-RAG's single-field queries derive from QR-LLM outputs generated from the complete original input, implicitly encoding context from all fields. The experimental implications of this distinction are addressed in Section 5.1.6. The *Title-Only* strategy constructs a single-line query from the Stage 2 transformed job title. The *Tasks-Only* strategy constructs a block query from the 10 hypothetical job-task sentences produced by Stage 3.

Conversely, the two *multi-field formulations* aggregate multiple data dimensions and are not subject to the information asymmetry, as both systems draw on the complete scope of available input data. The *Title+Tasks* strategy combines the Stage 2 transformed title with the 10 hypothetical job-task sentences into a multi-field

query. Finally, the *Schema-Data* strategy integrates the retained Stage 2 occupational attributes—excluding location and multi-role data to prevent retrieval degradation (Section 4.5.1.3)—alongside the hypothetical job tasks into a structured block. Each attribute field appears only when the corresponding Stage 2 output is non-empty, so the query's breadth adapts to the richness of the extracted content. Before assembling that query, the system converts the requirement level from its numeric code to the corresponding KldB label to provide descriptive semantic context for retrieval. The four labels are, in ascending complexity: (1) "Helfer- und Anlerntätigkeiten"—unskilled and semi-skilled tasks, (2) "Fachlich ausgerichtete Tätigkeiten"—skilled tasks, (3) "Komplexe Spezialistentätigkeiten"—complex specialist tasks, and (4) "Hoch komplexe Tätigkeiten"—highly complex tasks.

4.6 Summary

This chapter presents the implementation of the two RAG-based occupation-coding systems evaluated in this thesis—Naive-RAG and QR-RAG—both operating on the preprocessed DZHW survey data and querying a shared knowledge base built from the official KldB 2010 documentation. Experimental isolation is the central design principle: query preparation is the sole variable separating the two approaches; the knowledge base, retrieval stage, and LLM-based classification stage are identical across both. Hardware constraints—a dual RTX 3060 workstation with 24 GiB total VRAM, operated locally under DZHW data-privacy regulations—shaped the entire inference stack: Ollama was chosen for its consumer-grade GPU support and built-in concurrency, while 4-bit weight quantization and 8-bit key-value-cache quantization reduce VRAM requirements by over 50% with minimal quality degradation, enabling comprehensive evaluation within local resource constraints.

The shared knowledge base comprises 1,300 task-description documents extracted from official KldB PDFs via Docling and indexed in a Chroma vector store with HNSW-based cosine similarity search. Task descriptions were selected as the sole content field for three reasons: they likely provide the strongest semantic discriminator among available document sections; their task-oriented content was expected to create content overlap with the language respondents use in survey responses; and each list fits within the minimum EM context window without chunking, preserving every document as a coherent thematic unit. Nine separate vector-database instances—one per evaluated EM—ensure query and document vectors always share the same semantic space; documents are embedded once at system initialization and persisted to disk for reuse across all experimental runs.

Both approaches share identical retrieval and classification stages, with query preparation as the only upstream difference. The retrieval stage embeds the formulated query using the same dual-encoder EM instance that produced the corresponding vector store, matches it against the knowledge base via cosine similarity, and returns the top 20 candidates—a depth established as optimal in the component-level experiments—passing all candidates unconditionally to the classifier. The classification stage employs the optimal query-rewriting LLM in a dual role as shared classifier, applying a structured German-language prompt that pairs each retrieved candidate's KldB code with its task descriptions against the occupational data of

the instance being classified, with output constrained to the 5-digit code to enable deterministic parsing.

Naive-RAG feeds original survey job title and task fields directly into the query-formulation stage, using three strategies—Title-Only, Tasks-Only, and Title+Tasks—to test whether unmodified respondent language suffices for accurate retrieval without transformation. QR-RAG prepends a three-stage LLM-based query-rewriting pipeline upstream of the shared retrieval and classification stages. Stage 1 corrects surface-level defects in the original survey data—encoding errors, inconsistent capitalization, and unknown abbreviations—without altering semantic content. Stage 2 extracts structured occupational attributes aligned with the KldB taxonomy—including occupational title, industry, specialization, work setting, requirement level, and hierarchy role—and relocates misplaced content between fields. Stage 3 generates ten hypothetical task sentences in the formal register of KldB documents, directly targeting query–document alignment with the task-oriented knowledge base. A fourth QF exclusive to QR-RAG, Schema-Data, combines Stage 2 structured attributes with Stage 3 hypothetical tasks, designed to provide the richest available retrieval signal. To keep this three-stage pipeline feasible under consumer-grade hardware constraints, a concurrent batch-prompting architecture implemented via a shared LangChain Runnable Sequence processes multiple queries in parallel across all inference stages.

One limitation constrains the interpretability of this chapter's implementation. Stage 1 of the query-rewriting pipeline was not ablated independently: because it performs surface-level correction rather than semantic enrichment, no separate QF using Stage 1 output was defined, and its isolated contribution to retrieval performance therefore cannot be quantified. The following chapter evaluates both systems across nine EMs, six query-rewriting LLMs, and four stratified datasets to answer the thesis's research questions.

Listing 4.16: QR-RAG Query Formulation Templates. Structural templates for all four QR-RAG query formulations. Schema-Data's attribute fields are included on a per-field basis—each field appears only when the corresponding Stage 2 output is non-empty—so query length and breadth adapt to the richness of the extracted content.

```
##### Single-Field Query Formulations

# 1. Title-Only Template
Suche Berufsaufgaben für: {job_title_transformed} # from Stage 2

# 2. Tasks-Only Template
Suche Berufsaufgaben für:
---
- {hypothetical_task_1}                          # from Stage 3
- {hypothetical_task_2}
- ...
- {hypothetical_task_10}
---

##### Multi-Field Query Formulations

# 3. Title+Tasks Template
Suche Berufsaufgaben für:
---
{job_title_transformed}:                         # from Stage 2
- {hypothetical_task_1}                          # from Stage 3
- {hypothetical_task_2}
- ...
- {hypothetical_task_10}
---

# 4. Schema-Data Template
Suche Berufsaufgaben für:
---
Berufstitel: {job_title_transformed}             # from Stage 2, if non-empty
Spezialisierung: {specialization}                # from Stage 2, if non-empty
Industrie: {industry}                            # from Stage 2, if non-empty
Arbeitsumfeld: {work_setting}                    # from Stage 2, if non-empty
Arbeitgebertyp: {employer_type}                  # from Stage 2, if non-empty
Anforderungsniveau: {requirement_level_label}    # from Stage 2, if non-empty
Hierarchieebene: {hierarchy_role}                # from Stage 2, if non-empty
Berufsaufgaben:
- {hypothetical_task_1}                          # from Stage 3
- {hypothetical_task_2}
- ...
- {hypothetical_task_10}
---
```

Listing 4.17: QR-RAG Query Formulation Examples. The four templates instantiated using the continuous running example from Listing 4.10, rendering each formulation's output concrete. The *Arbeitgebertyp* field is conditionally omitted from the Schema-Data query because the running-example input contains no employer-type information.

```
##### Single-Field Query Formulations

# 1. Title-Only Example
Suche Berufsaufgaben für: Ernährungstherapeut

# 2. Tasks-Only Example
Suche Berufsaufgaben für:
---
- Ernährungspläne für Patienten mit Essstörungen im stationären Bereich
  individuell erstellen und anpassen
- Ernährungsberatung in Einzel- und Gruppengesprächen durchführen und
  dokumentieren
- ...
---

##### Multi-Field Query Formulations

# 3. Title+Tasks Example
Suche Berufsaufgaben für:
---
Ernährungstherapeut:
- Ernährungspläne für Patienten mit Essstörungen im stationären Bereich
  individuell erstellen und anpassen
- Ernährungsberatung in Einzel- und Gruppengesprächen durchführen und
  dokumentieren
- ...
---

# 4. Schema-Data Example
Suche Berufsaufgaben für:
---
Berufstitel: Ernährungstherapeut
Spezialisierung: Ernährungstherapie
Industrie: Gesundheitswesen
Arbeitsumfeld: Klinik, stationär
Anforderungsniveau: Komplexe Spezialistentätigkeiten
Hierarchieebene: Fachkraft
Berufsaufgaben:
- Ernährungspläne für Patienten mit Essstörungen im stationären Bereich
  individuell erstellen und anpassen
- Ernährungsberatung in Einzel- und Gruppengesprächen durchführen und
  dokumentieren
- ...
---
```

5. Evaluation

This chapter presents a comprehensive evaluation of our unsupervised RAG-based occupation coding on the KldB 2010 using German free-text survey data from our DZHW evaluation datasets, addressing our research questions defined in Section 1.3.4. We evaluate two implementations: Naive-RAG, which operationalizes the standard RAG approach; and QR-RAG, which operationalizes the query-rewriting-enhanced RAG approach as defined in Section 1.3.2. Throughout this chapter, we use the implementation and approach names interchangeably, as each implementation directly realizes its corresponding approach. Our evaluation systematically characterizes how individual architectural components—embedding models (EMs), QR-LLMs, and query formulations (QFs)—influence retrieval and classification performance, enabling us to determine optimal deployment configurations and quantify the effectiveness of LLM-based query rewriting (QR) across diverse information-richness conditions and KldB 2010 hierarchy levels.

The evaluation proceeds through five complementary sections. Section 5.1 establishes the experimental framework, detailing the evaluation datasets, knowledge base, evaluation metrics, language models, and query formulations. Section 5.2 addresses SQ-1 by characterizing Naive-RAG's retrieval performance, isolating the effects of individual embedding models. Section 5.3 addresses SQ-2 by evaluating QR-RAG's retrieval effectiveness, isolating the effects of individual QR-LLMs. Sections 5.4 and 5.5 address SQ-3 through system-level comparisons using the optimal configurations identified in SQ-1 and SQ-2, first examining how information richness moderates QR benefits for retrieval, then determining how effectively retrieval performance translates into downstream classification performance across KldB hierarchy levels. Collectively, these investigations answer the main research question by characterizing the extent and nature of performance differences between both approaches, identifying the conditions under which any improvements or trade-offs emerge, and deriving deployment guidance for practitioners and researchers implementing unsupervised RAG-based occupation-coding systems.

5.1 Experimental Design

This section establishes the experimental framework for evaluating Naive-RAG and QR-RAG across the three subquestions defined in Section 1.3.4. All experiments were conducted on a custom-built workstation with dual NVIDIA GeForce RTX 3060 GPUs providing 24 GiB total VRAM, reflecting the consumer-grade hardware constraints typical of smaller research institutions and survey organizations (detailed specifications and software configuration in Appendix A.2). Our experimental design systematically isolates and characterizes the influence of key architectural components on retrieval and classification performance, enabling us to determine optimal deployment configurations and quantify the effectiveness of query rewriting across diverse information-richness conditions and KldB hierarchy levels.

The experimental framework comprises six foundational components: (i) evaluation datasets that operationalize input complexity through stratified job-title length (Section 5.1.1); (ii) a standardized knowledge base ensuring controlled comparison between both approaches (Section 5.1.2); (iii) established metrics for retrieval and classification effectiveness (Section 5.1.3); (iv) nine embedding models with varying parameter counts and architectures (Section 5.1.4); (v) six high-parameter QR-LLMs with diverse architectures (Section 5.1.5); and (vi) four query formulations representing varying context density (Section 5.1.6). Together, these components generate the experimental conditions required to address each subquestion while controlling for confounding factors through systematic aggregation.

5.1.1 Evaluation Datasets

To combine real-world authenticity with systematic variation in input complexity—the first dimension of information richness (Section 1.3.3)—we utilized four complementary datasets derived from preprocessed DZHW graduate survey data. Each dataset contains free-text job descriptions—comprising non-standardized job titles and task fields—paired with ground-truth KldB codes assigned by expert coders. With sizes ranging from 856 to 1,088 instances, these datasets enable statistically meaningful comparisons while remaining computationally feasible for the costly LLM inference required by our RAG approaches.

We operationalized input complexity using job-title length as a proxy. The LJT and SJT datasets ($n = 856$ each) represent the extremes of this complexity spectrum, consisting of the longest and shortest job titles drawn from strata selected to exhibit sufficient internal length variation. Complementing these extremes, RVD ($n = 856$) serves as a random baseline drawn from these same variance-constrained strata, while RAD ($n = 1{,}088$) ensures complete coverage of the 1,088 distinct codes present in the preprocessed data through unconstrained random sampling. This stratified design enables us to assess RAG robustness across diverse levels of input complexity while preserving the full taxonomic breadth of the underlying survey data. Detailed dataset construction procedures are provided in Section 3.6.1.

To ensure realistic evaluation conditions, the datasets retain the authentic data-quality challenges inherent in free-text survey responses, including unknown abbreviations, inconsistent capitalization, and expression variability (Section 3.6.2, Table 3.3). Preserving these artifacts tests whether our RAG approaches can effectively handle the noisy inputs characteristic of operational environments.

5.1.2 Retrieval Knowledge Base

Both Naive-RAG and QR-RAG retrieve from an identical knowledge base to isolate the effects of query rewriting from knowledge-base variations, ensuring controlled comparison. The knowledge base contains 1,300 job-task descriptions derived from the official German KldB 2010 documentation. These task descriptions were curated by the taxonomy's authors and are deliberately formal and broad, defining general occupational categories (e.g., "software developers") rather than specific job positions (e.g., "backend engineer at DeepMind"). In contrast, survey respondents in our evaluation datasets describe their individual jobs using informal, everyday language. This semantic gap tests whether our RAG approaches can bridge mismatched vocabularies spanning two dimensions: register mismatch (formal vs. colloquial) and granularity mismatch (category-oriented vs. position-oriented), as defined in Section 1.1.2. Detailed knowledge-base construction procedures are provided in Section 4.2.

5.1.3 Retrieval and Classification Metrics

We evaluate our RAG approaches using distinct metrics for retrieval and classification stages, adhering to established standards in information retrieval and ML [Manning et al., 2008; Sokolova and Lapalme, 2009].

Retrieval performance is evaluated at three depths ($k = 5, 10, 20$) using two complementary metrics. Recall@k measures retrieval coverage by computing the proportion of relevant documents retrieved within the top-k results [Manning et al., 2008]. Since our ground truth contains exactly one relevant document per query, Recall@k simplifies to a binary metric (1.0 if the target is found, 0.0 otherwise). While Recall@k indicates whether the correct document was retrieved, NDCG@k (Normalized Discounted Cumulative Gain (NDCG)) reveals how well it is ranked within the list. NDCG@k assigns higher scores when relevant documents appear earlier, discounting contributions logarithmically by rank position [Järvelin and Kekäläinen, 2002]. With our binary relevance judgments (relevant = 1, not relevant = 0), NDCG@k strictly reflects the rank position of the correct document rather than varying degrees of relevance.

Classification performance is evaluated using Cohen's Kappa and Accuracy, computed separately across all five KldB hierarchy levels (1-digit to 5-digit codes). Cohen's Kappa assesses agreement beyond chance [Cohen, 1960], while Accuracy provides a direct measure of classification correctness [Sokolova and Lapalme, 2009]. This multi-level evaluation allows us to diagnose how performance degrades with increasing taxonomic granularity—from broad occupational areas (1-digit) to specific job categories (5-digit)—enabling us to determine whether the benefits of QR-RAG generalize across the full KldB hierarchy.

5.1.4 Embedding Models

Multilingual text embedding models serve as a foundational retrieval component in our RAG systems, transforming text into semantic vector representations that enable semantic search. To effectively retrieve relevant documents from the knowledge base, these models must capture the nuances of German occupational terminology

and map colloquial queries to formal, taxonomic task descriptions—the register and granularity mismatches defining the semantic gap (Section 1.1.2). Meeting these linguistic requirements necessitates models with robust multilingual support and substantial internal knowledge.

As established in Section 1.1.4, multilingual text EMs utilize dual-encoder architectures that enable efficient retrieval by allowing pre-computation of knowledge-base embeddings and fast semantic similarity matching at query time [Karpukhin et al., 2020]. Unlike autoregressive LLMs, these dual-encoder models produce fixed-length vectors without text generation or reasoning capabilities. This computationally less demanding architecture enabled us to deploy most models at full 16-bit floating-point (FP16) or 16-bit brain floating-point (BF16) precision while fitting comfortably within VRAM. However, for the exceptionally large Qwen3-Embedding (7.6B parameters), we utilized Q4_K_M quantization to ensure deployment feasibility on our hardware, as full-precision execution would exceed VRAM capacity.

Based on these considerations, we selected nine multilingual text EMs available through Ollama, representing diverse state-of-the-art architectures (e.g., BERT-based and GPT-like encoders). Table 5.1 presents their technical specifications, including parameter count, precision level, and context capacity.

Table 5.1: Specifications of Selected Multilingual Text Embedding Models. Technical details of the nine multilingual text embedding models (161M–7.6B parameters) selected for evaluation. To ensure deployment feasibility on limited hardware, the large Qwen3-Embedding model is quantized to Q4_K_M, whereas smaller models use standard FP16/BF16 precision. The *Ollama Model Digest* serves as a unique checksum for reproducibility.

Embedding Model	Ollama Model Digest	Precision Level	Parameter Count	Max Context Window
Qwen3-Embedding	64b933495768	Q4_K_M	7.6B	40,960
Snowflake-Arctic-Embed2	5de93a84837d	FP16	567M	8,192
BGE-M3	790764642607	FP16	567M	8,192
Multilingual-E5-Large	fc1b80b14a3d	FP16	559M	512
MxBai-Embed-Large	468836162de7	FP16	334M	512
EmbeddingGemma	85462619ee72	BF16	308M	2,048
Granite-Embedding	1a37926bf842	FP16	278M	512
Paraphrase-Multilingual	ba13c2e06707	FP16	278M	512
Jina-v2-Base-DE	f9b10d7e24af	FP16	161M	8,192

To further optimize retrieval performance, we leveraged specific execution capabilities of the selected models. First, we utilized the instruction-finetuned capabilities of models where applicable, using their default task-specific prefixes for queries and documents to guide the embedding process [Su et al., 2023]. However, initial experiments indicated that the default prefixes for Qwen3-Embedding and MxBai-Embed-Large degraded performance; consequently, we executed these models without prefixes. Second, we configured all models to utilize their maximum supported context window, ensuring that even the most extensive knowledge-base task descriptions were processed without truncation. Table 5.2 details the complete execution configuration.

Table 5.2: Execution Parameters of Selected Multilingual Text Embedding Models. Configuration details for deployment. *Task Aware* indicates native support for instruction-finetuned embedding. *Prefixes* denote specific text strings prepended to inputs; models marked with (*) support instructions but were executed without prefixes to avoid empirically observed performance degradation. *Context Window* defines the maximum token limit utilized during experiments.

Embedding Model	Task Aware	Query Prefix	Document Prefix	Context Window
Qwen3-Embedding*	yes			40,960
Snowflake-Arctic-Embed2	yes	query:		8,192
BGE-M3	—			8,192
Multilingual-E5-Large	yes	query:	passage:	512
MxBai-Embed-Large*	yes			512
EmbeddingGemma	yes	task: search result | query:	title: none | text:	2,048
Granite-Embedding	—			512
Paraphrase-Multilingual	—			512
Jina-v2-Base-DE	—			8,192

We utilize all nine models to systematically characterize Naive-RAG's retrieval performance in SQ-1 (Section 5.2), enabling an empirical assessment of how individual models perform for German occupational data. These models are also employed in SQ-2 (Section 5.3) to evaluate QR-RAG's retrieval performance, where results are aggregated across all embedding models and evaluation datasets to isolate the impact of individual QR-LLMs. Subsequently, we identify the optimal embedding model from SQ-1, which is used for system-level experiments in SQ-3 (Sections 5.4 and 5.5). This strategy ensures we utilize the highest achievable retrieval quality when comparing system performance between Naive-RAG and QR-RAG.

5.1.5 Query-Rewriting LLMs

While embedding models enable retrieval of relevant documents, their performance can be substantially influenced by query quality. QR-LLMs serve as the core generative component in QR-RAG, transforming colloquial survey responses—utilized as search queries—into formalized, task-enriched descriptions aligned with the KldB documentation. As established in Section 1.1.4, this QR mechanism addresses the semantic gap by enriching terse survey responses or clarifying ambiguous ones, thereby improving query–document alignment [Koo et al., 2024; Ma et al., 2023]. This transformation entails correcting orthography, extracting occupational details, and generating hypothetical job tasks—complex linguistic operations requiring models with robust German-language support and substantial occupational domain knowledge.

To meet these demands, we prioritized multilingual models with high parameter counts (14B–32B), leveraging their typically broader domain knowledge and superior instruction-following capabilities. However, to make execution feasible on our hardware, we adopted a specific optimization strategy: maximizing model size to ensure knowledge capacity while standardizing on 4-bit quantization (Q4_K_M) for all models, leveraging quantization techniques that enable LLM deployment on

consumer-grade hardware [Husom et al., 2025]. This trade-off allowed us to deploy 14B+ parameter models locally that would otherwise exceed VRAM capacity at higher precision.

Guided by these considerations, we selected six multilingual LLMs available through Ollama. To ensure robust evaluation, we included models representing various state-of-the-art architectures including sparsely-activated MoE models [Yang et al., 2025], with capabilities ranging from standard instruction-following models (e.g., Phi-4) to variants with reasoning capabilities (e.g., DeepSeek-R1, Qwen3) [Guo et al., 2025; Patil and Jadon, 2025]. This selection includes parameter variations within the same model family (Qwen3-14B vs. Qwen3-32B), enabling analysis of how model complexity impacts retrieval performance. Table 5.3 presents their technical specifications.

Table 5.3: Specifications of Selected Query-Rewriting LLMs. Technical details of the six high-parameter (14B–32B) models used for query rewriting. All models are uniformly deployed at 4-bit quantization (Q4_K_M) to maximize parameter count within available VRAM constraints. The selection includes various state-of-the-art architectures (standard instruction-following and sparsely-activated MoE models) with different capabilities (standard vs. reasoning-capable models such as DeepSeek-R1 and Qwen3). The *Ollama Model Digest* serves as a unique checksum for reproducibility.

QR-LLM	Ollama Model Digest	Precision Level	Parameter Count	Max Context Window
Reasoning-Capable Models				
Qwen3-32B	030ee887880f	Q4_K_M	32B	40k
Magistral-24B	27bcbbf6d324	Q4_K_M	24B	39k
Qwen3-14B	bdbd181c33f2	Q4_K_M	14B	40k
DeepSeek-R1-14B	c333b7232bdb	Q4_K_M	14B	128k
Standard Models				
Qwen3-30B-MoE	19e422b02313	Q4_K_M	30B	256k
Phi4-14B	ac896e5b8b34	Q4_K_M	14B	16k

Executing these high-parameter models locally introduces significant computational overhead due to expensive autoregressive token generation. To maximize processing efficiency, we implemented concurrent batch prompting in QR-RAG (Section 4.5.2). This approach balances concurrency levels against each model's memory footprint. We adjusted concurrency based on VRAM availability but maintained a constant batch size of five samples per prompt. This batch size aligns with findings by Cheng et al. [2023], who identified that batch sizes of five to six samples offer optimal trade-offs between token throughput and task performance.

Initial experiments revealed that activating reasoning mode increased execution time by 3–6× and significantly expanded context-window usage due to verbose reasoning trace generation. To avoid excessive latency and ensure comparable evaluation conditions, we configured all reasoning-capable models to execute in standard mode (reasoning disabled). Table 5.4 details the execution configuration for each QR-LLM.

Table 5.4: Execution Parameters of Selected Query-Rewriting LLMs. Configuration details for deployment. *Concurrency* denotes the number of parallel requests, adjusted per model to fit VRAM limits. *Batch Size* is standardized to five samples per prompt to balance throughput and quality. *Reasoning* is explicitly disabled for reasoning-capable models to prevent excessive latency. *Context Window* defines the maximum token limit utilized during experiments.

QR-LLM	Concurrency	Batch Size	Reasoning	Context Window
Reasoning-Capable Models				
Qwen3-32B	3	5	off	5,400
Magistral-24B	6	5	off	5,400
Qwen3-14B	10	5	off	5,400
DeepSeek-R1-14B	8	5	off	5,400
Standard Models				
Qwen3-30B-MoE	10	5	—	5,400
Phi4-14B	8	5	—	5,400

We utilize all six models to systematically characterize QR-RAG's retrieval performance in SQ-2 (Section 5.3), enabling empirical assessment of how individual models transform query quality through rewriting for German occupational data and thereby influence downstream retrieval effectiveness while balancing deployment feasibility. Subsequently, we identify the optimal QR-LLM, which is used for system-level experiments in SQ-3 (Sections 5.4 and 5.5) as the query rewriter for QR-RAG and as the LLM-based classifier for both RAG approaches.

5.1.6 Query Formulations

The effectiveness of a retrieval system depends largely on the content and structure of the search query. To systematically evaluate how different levels of occupational context influence retrieval performance, we employ four distinct QFs as experimental conditions: Title-Only, Tasks-Only, Title+Tasks, and Schema-Data. These formulations range from low density structures (relying solely on job titles) to high density structures (integrating LLM-extracted occupational attributes), operationalizing the context-density dimension of information richness defined in Section 1.3.3.

Table 5.5 details how these four formulations are instantiated across both RAG systems. A key distinction arises in the single-field formulations due to our implementation of query rewriting. We illustrate this distinction using the Title-Only formulation. In Naive-RAG, the query is simply the raw job title extracted directly from the evaluation data. In QR-RAG, however, this query is a rewritten version generated by a QR-LLM that processed the entire input description—including job tasks—to generate the output. Consequently, while Naive-RAG's single-field formulations use the original input fields in isolation, QR-RAG constructs its queries by synthesizing all available input fields. This creates an information asymmetry: QR-RAG's single-field queries implicitly contain context from the full input, whereas Naive-RAG's do not. This asymmetry disappears in multi-field formulations (e.g., Title+Tasks), where both systems utilize the full scope of the original input data. Therefore, we utilize multi-field formulations for unbiased cross-system comparisons

in SQ-3, while reserving single-field formulations to isolate the specific contribution of each field to overall retrieval performance. We explicitly account for this contextual information synthesis in QR-RAG's single-field queries during analysis, treating it as an inherent feature of the rewriting process rather than an experimental confound.

Table 5.5: Selected Query Formulations from Naive-RAG and QR-RAG. The four query formulations across both RAG approaches used for evaluation. *Context Density* indicates the structural complexity of the query. Naive-RAG formulations use raw fields extracted directly from the preprocessed evaluation data, whereas QR-RAG formulations utilize rewritten text generated by a QR-LLM. The hypothetical job tasks consist of 10 LLM-generated sentence-level descriptions per query instance. Note that Schema-Data is exclusive to QR-RAG as it relies on LLM-extracted occupational attributes (e.g., industry and specialization).

Query Formulation	Naive-RAG (Source: Evaluation Data)	QR-RAG (Source: LLM-Rewritten Data)	Context Density
Title-Only	original job title	transformed job title (QR Stage 2)	low
Tasks-Only	original job tasks 1 and 2	hypothetical job tasks (QR Stage 3)	high
Title+Tasks	original job title and tasks	transformed title and hypothetical tasks	high
Schema-Data	—	transformed title and hypothetical tasks plus extracted occupational attributes (QR Stage 2)	very high

This experimental design operationalizes the two dimensions of information richness defined in Section 1.3.3. First, the context-density dimension tests the value of adding occupational context (e.g., determining whether appending associated tasks to a title improves retrieval). Second, the input-complexity dimension tests the impact of input length, which varies systematically across our four evaluation datasets as detailed in Section 5.1.1. By crossing these four formulations with these four datasets in SQ-3, we can determine whether the benefits of query rewriting generalize across diverse levels of information richness.

5.2 SQ-1: Naive-RAG Retrieval Performance

Following the experimental framework established in Section 5.1, this section addresses SQ-1 by evaluating the retrieval stage of Naive-RAG. Our primary focus is to determine how individual embedding models influence retrieval effectiveness by transforming German free-text survey responses and KldB descriptions into semantic vector representations, thereby assessing their capabilities to bridge the semantic gap between colloquial input and formal taxonomic content. We isolate embedding-model effects by controlling for confounding factors through systematic aggregation.

Our analysis covers 81 experimental settings generated by the cross-product of the nine embedding models (Table 5.1), three query formulations (Table 5.5), and three retrieval depths ($k = 5, 10, 20$). To characterize embedding-model performance while controlling for dataset influences, we report Recall@k and NDCG@k results aggregated over the four DZHW evaluation datasets (Section 5.1.1). Figures 5.1 and 5.2 display the retrieval performance across all experimental settings, with the underlying numerical data provided in Tables A.4 and A.5.

Our analysis proceeds along three complementary lines of investigation: (i) identifying the optimal embedding model for deployment; (ii) determining which query formulation (original survey input) provides the dominant retrieval signal; and (iii) establishing the optimal retrieval depth that balances candidate coverage against downstream processing costs. The optimal embedding model and retrieval depth identified here inform deployment decisions for SQ-3 experiments, with the retrieval depth choice validated by parallel findings from SQ-2 (Section 5.3).

5.2.1 Performance Tiers Reveal Only Top-Tier EMs Are Viable

We anticipated that embedding models would differ in retrieval performance due to architectural and training differences, reflecting known substantial performance variation across EMs for retrieval tasks [Thakur et al., 2021]. However, we remained agnostic about whether the performance variation observed in retrieving KldB documents using German free-text survey responses would manifest as gradual optimization differences or discrete tiers separating viable from non-viable models.

Our measurements revealed substantial performance stratification that manifests as discrete tiers rather than gradual optimization differences (Figures 5.1 and 5.2). At $k = 20$ using the Title+Tasks formulation, Recall ranged from 0.148 (EmbeddingGemma) to 0.767 (Qwen3-Embedding)—an 81% performance deficit for the worst model relative to the best. The models formed three discrete tiers: a top tier (Qwen3-Embedding, Snowflake-Arctic-Embed2) achieving 0.72–0.77 Recall; a mid tier (Multilingual-E5-Large, BGE-M3, Granite-Embedding, Jina-v2-Base-DE) achieving 0.51–0.62 Recall; and a bottom tier (MxBai-Embed-Large, Paraphrase-Multilingual, EmbeddingGemma) achieving 0.15–0.37 Recall. Relative to the best-performing model (Qwen3-Embedding, 0.767), mid-tier models showed 19–34% deficits while bottom-tier models showed 52–81% deficits. This 3-tier structure remained stable across all retrieval depths and query formulations for both metrics, with only minor within-tier rank shifts.

The tier-based stratification reflects different model capabilities to address the semantic gap challenges established in Section 1.1.2. Models must bridge two dimensions: register mismatch (colloquial survey responses vs. formal KldB definitions) and granularity mismatch (position-oriented vs. category-oriented job descriptions), requiring robust understanding of German occupational terminology across both vocabularies. Additionally, models must handle authentic data-quality issues such as expression variability and unknown abbreviations (Table 3.3), demanding contextual interpretation beyond simple lexical matching. The dramatic performance gaps suggest that top-tier models possess stronger German-language capability and richer internal knowledge of the occupational domain, enabling them to decode domain-specific nuances and map informal descriptions to formal counterparts—capabilities that lower-tier models fundamentally lack. Because documents missed during retrieval create an irrecoverable bottleneck for the downstream classifier, only top-tier models provide sufficient candidate coverage for deployment.

These findings answer the embedding-model dimension of SQ-1, revealing three key insights. First, performance stratification manifests as discrete tiers with dramatic spread (81% deficit from bottom to top), not gradual optimization differences. This

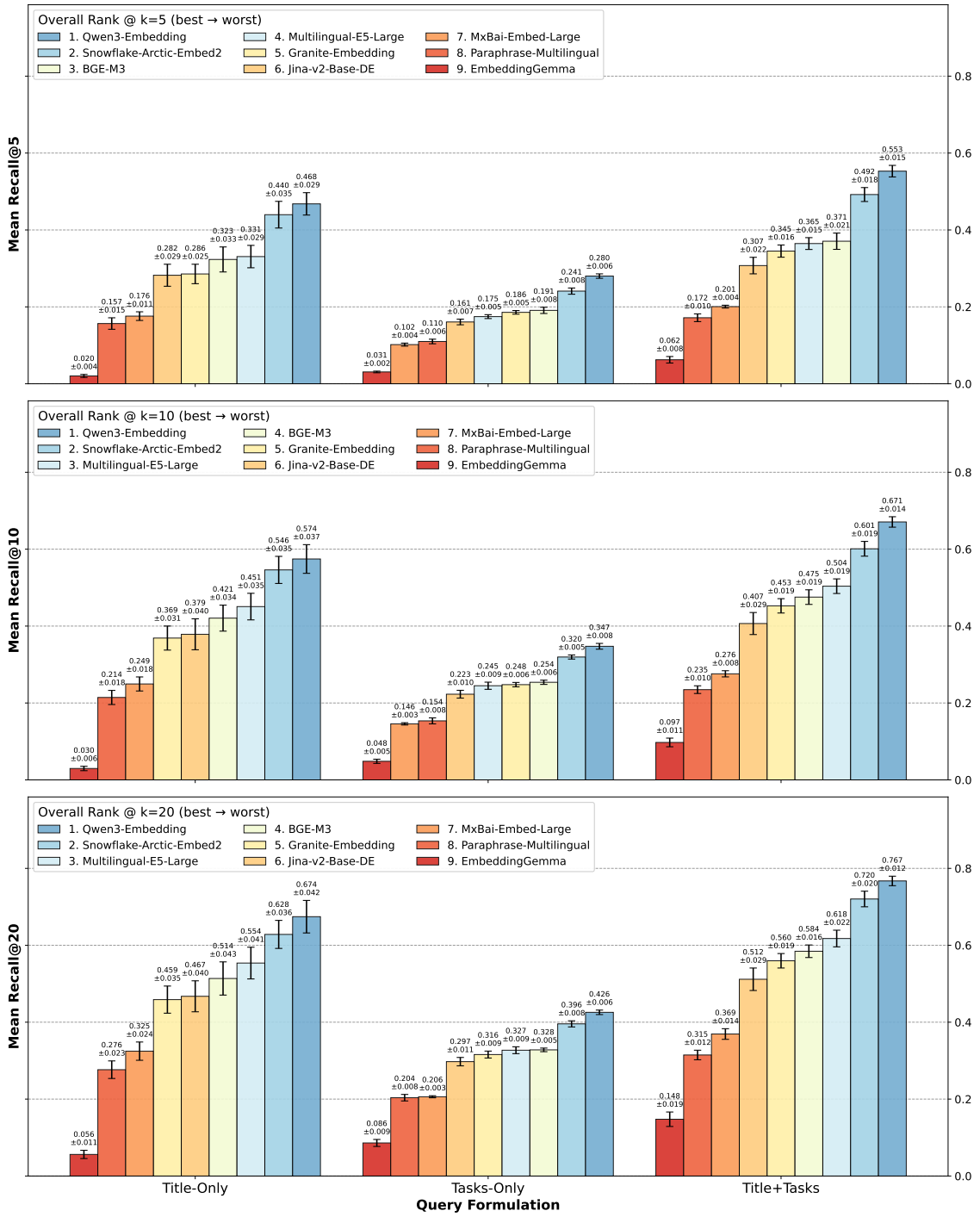


Figure 5.1: Impact of Embedding Models on Recall Performance in Naive-RAG. EMs form three discrete performance tiers with Qwen3-Embedding achieving highest mean Recall (0.767 at $k = 20$ with Title+Tasks), job titles providing the dominant retrieval signal (Title-Only: 88% of Title+Tasks performance), and $k = 20$ capturing substantial long-tail gains (+9.6pp over $k = 10$). The figure displays aggregated Recall@k results covering 81 experimental settings (9 EMs × 3 QFs × 3 retrieval depths), averaged across four evaluation datasets. Panels correspond to retrieval depths ($k = 5, 10, 20$). Within each panel, bars are grouped by QFs and sorted by EM performance (worst to best, left to right). Numeric annotations above bars represent the mean ± standard error of the mean (SEM) to quantify uncertainty across datasets. Legends rank EMs by overall performance at each k-level (best to worst, top-left to bottom-right). Color coding is consistent across all panels for direct comparison.

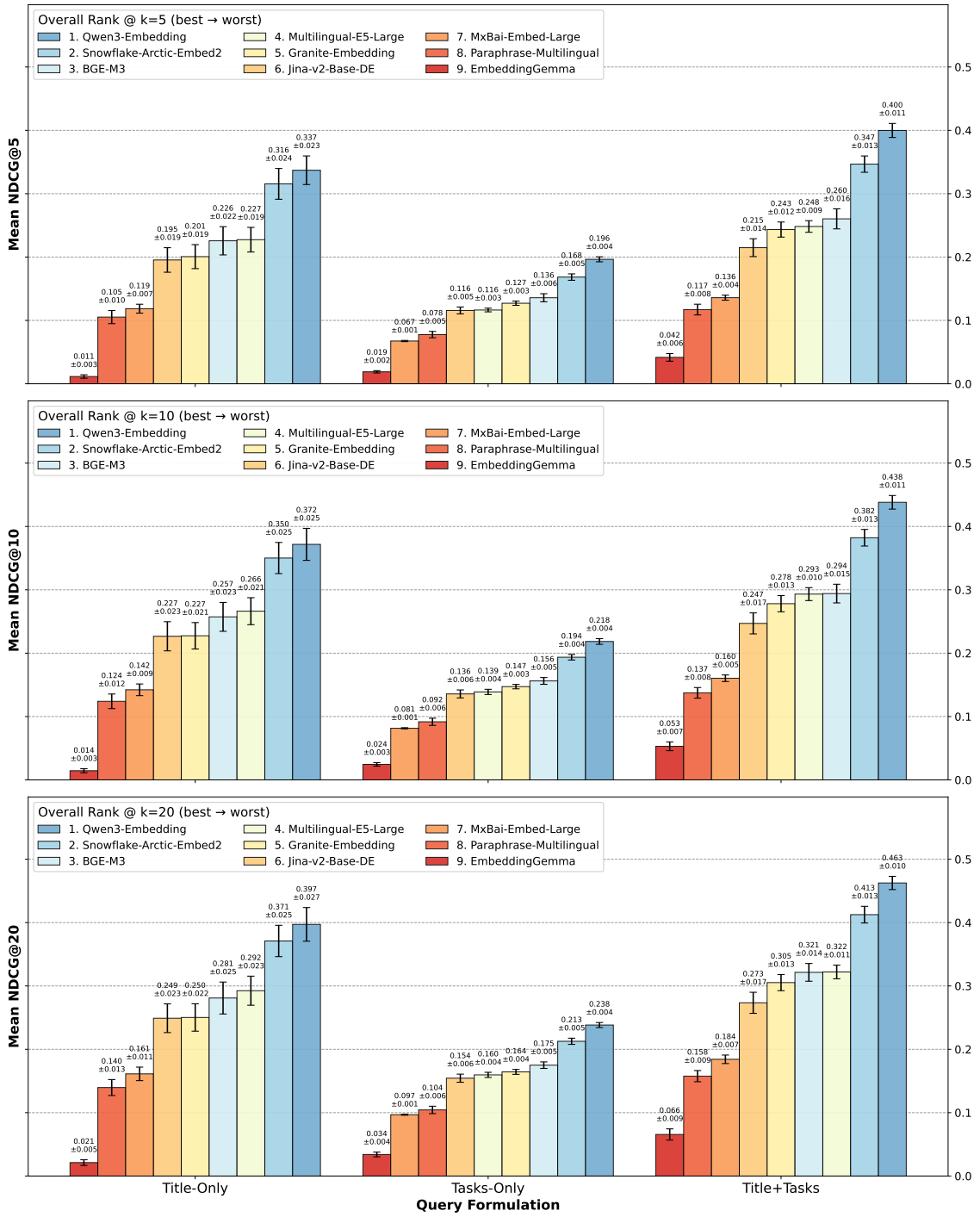


Figure 5.2: Impact of Embedding Models on NDCG Performance in Naive-RAG. NDCG metrics confirm the Recall patterns: 3-tier EM stratification with Qwen3-Embedding achieving highest ranking quality, job-title dominance, and $k = 20$ long-tail gains. The figure displays aggregated NDCG@k results covering 81 experimental settings (9 EMs × 3 QFs × 3 retrieval depths), averaged across four evaluation datasets. Panels correspond to retrieval depths ($k = 5, 10, 20$). Within each panel, bars are grouped by QFs and sorted by EM performance (worst to best, left to right). Numeric annotations above bars represent the mean ± standard error of the mean (SEM) to quantify uncertainty across datasets. Legends rank EMs by overall performance at each k-level (best to worst, top-left to bottom-right). Color coding is consistent across all panels for direct comparison.

establishes only top-tier models (Recall ≥ 0.72 at $k = 20$) as viable for deployment, while mid-tier models represent borderline viability and bottom-tier models represent fundamental failure. Second, tier boundaries likely reflect different capabilities to address semantic gap challenges (register/granularity mismatch, German occupational terminology, authentic data-quality issues), with top-tier models likely possessing stronger German-language capacity and richer occupational domain knowledge that lower-tier models fundamentally lack. Third, the optimal model selection within the top tier requires balancing retrieval quality against computational efficiency: Qwen3-Embedding achieves highest Recall (0.767) with 7.6B parameters, while Snowflake-Arctic-Embed2 achieves 94% of peak performance (0.720) with 13× fewer parameters (567M).

Having answered the embedding-model dimension of SQ-1, we apply these findings to our deployment strategy for SQ-3. We select Qwen3-Embedding as the optimal EM for classification experiments, prioritizing maximal candidate coverage. However, we retain all nine EMs across SQs 2 and 3 for retrieval experiments, enabling examination of potential interaction effects between EM tiers and QR effectiveness. Beyond our experimental decisions, these findings provide tier-based selection guidance for two external audiences. For survey organizations implementing occupation-coding systems, deployment decisions should balance computational resources against accuracy requirements: Snowflake-Arctic-Embed2 provides efficient inference for resource-constrained environments with acceptable 4.7pp deficit, while Qwen3-Embedding maximizes accuracy for organizations with adequate computational capacity. For researchers pursuing supervised fine-tuning of embedding models for German survey-based occupation coding, these findings identify which models provide the strongest foundation: starting with top-tier models as base architectures should yield superior results compared to prior work using standard BERT variants [Safikhani et al., 2023].

5.2.2 Job Titles Provide the Dominant Retrieval Signal

We anticipated that high-density QFs would outperform the low-density Title-Only formulation. Each high-density formulation incorporates typically two to three original job-task sentences from survey responses, whereas Title-Only relies solely on the original job title in Naive-RAG (Section 5.1.6). Our expectation rested on two premises: (i) the knowledge base contains task-oriented documents, suggesting closer structural and semantic alignment with task-based queries; and (ii) task fields provide more textual content for matching than brief job titles alone, which should enable more discriminative retrieval.

Our measurements revealed a clear performance dichotomy that contradicted these expectations (Figures 5.1 and 5.2). At $k = 20$ for the optimal Qwen3-Embedding model, Title+Tasks achieved the highest Recall at 0.767, followed closely by Title-Only at 0.674 (9.3pp deficit), while Tasks-Only trailed substantially at 0.426 (34.1pp deficit). Expressed as recovery rates relative to Title+Tasks, Title-Only achieved 88% while Tasks-Only achieved only 56%. This dichotomy remained stable across all retrieval depths and embedding models for both metrics, with only the bottom-tier EmbeddingGemma model exhibiting a reversal.

This performance pattern reveals that job titles alone capture the dominant retrieval signal in Naive-RAG, whereas task descriptions alone provide insufficient grounding.

This pattern parallels recent work on supervised German occupation coding, where automatic classification tools such as CASCOT similarly achieved substantially higher performance with job-title responses [Kononykhina et al., 2025], suggesting that title superiority generalizes across both retrieval and classification stages. We attribute the dominance of titles in our retrieval context to two data characteristics that outweigh the theoretical benefits of query–document alignment and greater content volume. First, job titles are mandatory and consistently populated across all evaluation instances, whereas task fields are optional and occasionally absent, providing broader coverage for title-based queries. Second, original task fields contain longer free-text responses with higher probability of noisy or inconsistent content that could interfere with retrieval, whereas job titles remain relatively short and concise.

These findings answer the context-density dimension of SQ-1, measured through QFs, revealing three key insights. First, job titles constitute the dominant retrieval signal in Naive-RAG, with Title-Only achieving 88% recovery rate compared to Tasks-Only's 56% (a 32pp gap), demonstrating that title-based queries provide the primary semantic anchor for occupation retrieval. Second, this dominance likely stems from data characteristics (mandatory coverage and conciseness) that outweigh theoretical advantages of query–document alignment and content volume, with longer task responses introducing greater noise and semantic drift compared to concise job titles. Third, the combined Title+Tasks formulation achieves optimal performance (0.767 Recall at $k = 20$), with task descriptions serving supplementary context that enhances but cannot replace the foundational title signal.

Having answered the query-formulation dimension of SQ-1, we apply these findings to our experimental design for SQ-3. In classification experiments, we employ Title+Tasks as the most effective QF for Naive-RAG. In retrieval experiments, we retain all three QFs as an experimental dimension representing context density, enabling systematic evaluation of information-richness effects on system-level performance. For research institutions and survey organizations implementing Naive-RAG systems, these findings provide clear data-collection guidance: prioritize high-coverage job-title fields, treating task descriptions as valuable supplementary context.

5.2.3 Long-Tail Gains Establish k = 20 as Optimal Depth

We anticipated that increasing retrieval depth would yield monotonic performance improvements with diminishing marginal returns, consistent with information retrieval principles [Manning et al., 2008]. However, we remained uncertain about the magnitude of long-tail gains—relevant documents residing beyond shallow retrieval depths—and whether these gains would justify the computational costs of deeper retrieval.

Our measurements confirmed monotonic improvement with diminishing marginal returns, while revealing substantial long-tail gains that justify deeper retrieval (Figures 5.1 and 5.2). For Qwen3-Embedding with Title+Tasks, Recall improved from 0.553 ($k = 5$) to 0.671 ($k = 10$, +11.8pp gain) and to 0.767 ($k = 20$, +9.6pp gain). Although the step from $k = 10$ to $k = 20$ spans twice the retrieval distance, the absolute gain declined by 18.6%, demonstrating declining marginal efficiency. This

progression pattern held robustly across all evaluated EMs and QFs for both metrics, with consistent long-tail gains even for lower-performing models.

The long-tail gains reflect the semantic heterogeneity of both survey responses and knowledge-base documents, which create varying degrees of query–document matchability. High-ranked candidates show clear lexical and structural alignment (e.g., formal job titles matching corresponding KldB entries), whereas lower-ranked candidates require deeper contextual interpretation to bridge multiple mismatches (e.g., inferring occupational category membership from position-specific task descriptions—granularity mismatch, or mapping informal survey language to technical taxonomy terminology—register mismatch). Expanding to $k = 20$ introduces computational costs—increased distractor noise, elevated token load for the LLM classifier, and higher inference latency—but these represent acceptable system constraints within our deployment assumptions. In contrast, relevant documents missed during retrieval create an irrecoverable bottleneck: the downstream classifier cannot recover what retrieval fails to surface. This fundamental asymmetry—retrieval failure imposes absolute performance ceilings while computational costs remain manageable—establishes $k = 20$ as optimal for Naive-RAG in our evaluation context, prioritizing maximum candidate coverage despite increased computational load.

These findings answer the retrieval-depth dimension of SQ-1, revealing three key insights. First, retrieval performance follows monotonic improvement with diminishing marginal returns ($0.553 \rightarrow 0.671 \rightarrow 0.767$, with marginal gains declining 18.6% from +11.8pp to +9.6pp), confirming standard information retrieval principles while demonstrating substantial long-tail gains persist at deeper depths. Second, the 9.6pp gain from $k = 10$ to $k = 20$ represents nearly 10% of total recallable documents—hard-to-match candidates requiring deeper contextual interpretation—establishing that shallow retrieval misses a non-trivial long tail. Third, $k = 20$ emerges as the optimal depth for our evaluation context because retrieval failure creates an irrecoverable bottleneck (documents missed cannot be recovered downstream), whereas computational costs from deeper retrieval (distractor noise, token load, latency) represent acceptable system constraints within our deployment assumptions prioritizing maximal coverage.

Having answered the retrieval-depth dimension of SQ-1, we apply these findings through a 2-stage validation strategy. For Naive-RAG, we establish $k = 20$ as the optimal retrieval depth based on long-tail coverage gains that justify computational costs. We validate whether this depth remains optimal for LLM-based query rewriting in SQ-2, examining whether rewritten queries alter the retrieval-depth trade-offs observed in Naive-RAG. Pending validation, we finalize $k = 20$ as the standard depth for system-level experiments in SQ-3. For practitioners implementing occupation-coding systems, these findings provide depth-selection guidance: prioritize $k = 20$ when maximizing coverage is critical and computational resources permit accepting increased token load and inference latency, but consider shallower depths ($k = 10$) for deployments with strict computational constraints accepting a 10% coverage deficit. For researchers developing retrieval systems for structured knowledge bases with semantic heterogeneity (e.g., occupational taxonomies, medical coding systems), these findings demonstrate that long-tail gains justify deeper retrieval when the cost of missed candidates (classification errors) outweighs computational costs.

5.3 SQ-2: QR-RAG Retrieval Performance

Following the characterization of Naive-RAG's retrieval performance across individual embedding models established in Section 5.2, this section addresses SQ-2 by evaluating the retrieval stage of QR-RAG. Our primary focus is to investigate how individual LLMs influence retrieval effectiveness by rewriting German free-text survey responses, thereby assessing their QR capabilities and computational efficiency. Paralleling the analytical approach in SQ-1, we isolate QR-LLM effects by controlling for confounding factors through systematic aggregation.

Our analysis covers 72 experimental settings comprising the cross-product of six QR-LLMs (Table 5.3), four QFs (Table 5.5), and three retrieval depths ($k = 5, 10, 20$). To characterize QR-LLM performance while controlling for EM and dataset influences, we report Recall@k and NDCG@k results aggregated over nine EMs (Table 5.1) and four DZHW evaluation datasets (Section 5.1.1). Figures 5.3 and 5.4 display the retrieval performance across all experimental settings, with the underlying numerical data provided in Table A.6. Additionally, Table 5.6 details the computational cost of the QR pipeline, quantifying execution latency and VRAM usage for each QR-LLM.

Our analysis proceeds along two complementary lines of investigation: (i) identifying the optimal QR-LLM that balances retrieval quality against computational efficiency; and (ii) determining which query formulation (rewritten survey input) provides the dominant retrieval signal. Regarding retrieval depth, performance patterns paralleled those in Naive-RAG (Section 5.2.3), confirming $k = 20$ as optimal for capturing long-tail candidates across both approaches. We therefore focus our analysis on $k = 20$, with results validated across all retrieval depths. The optimal QR-LLM identified here informs deployment decisions for SQ-3 experiments.

Table 5.6: Computational Cost of the Query-Rewriting Pipeline in QR-RAG. Qwen3-30B-MoE bypasses the quality–efficiency trade-off by achieving 99.4% of peak recall performance while executing 2.6× faster than Qwen3-32B (7.2h vs. 18.9h for 3,656 queries). The table reports execution latency and VRAM usage for each QR-LLM across the complete query-rewriting pipeline. Timings are reported in minutes per dataset and hours for the total workload (3,656 queries). *VRAM* indicates the allocated GPU memory footprint. *S1*, *S2*, and *S3* denote the sequential query-rewriting pipeline stages: content refinement, occupational attribute extraction, and hypothetical task generation. Cell shading highlights the extremes: green indicates the fastest execution per stage; red indicates the slowest. Note: The high latency in *RAD-S1* for Qwen3-14B results from reproducible execution stalls that necessitated manual intervention.

		LJT ($n = 856$)			**SJT** ($n = 856$)			**RVD** ($n = 856$)			**RAD** ($n = 1{,}088$)			
QR-LLM	**VRAM (GiB)**	**S1**	**S2 (m)**	**S3**	**S1**	**S2 (m)**	**S3**	**S1**	**S2 (m)**	**S3**	**S1**	**S2 (m)**	**S3**	**Total (h)**
Qwen3-32B	20.5	45.7	80.6	158.3	30.2	66.1	151.2	35.2	71.5	156.3	44.8	91.4	200.9	18.9
Qwen3-30B-MoE	19.6	16.0	31.0	61.4	11.0	25.1	59.2	12.6	26.9	60.4	15.4	33.5	76.5	7.2
Magistral-24B	21.4	23.7	39.0	65.3	73.4	32.2	59.6	19.5	35.0	61.8	24.0	44.0	77.2	9.2
Qwen3-14B	13.0	28.6	51.6	86.6	14.8	40.2	86.6	17.0	46.2	88.9	150.5	55.6	109.1	12.9
Phi4-14B	18.6	16.6	60.4	59.4	10.8	63.8	58.7	12.6	58.6	60.5	15.8	72.0	75.2	9.4
DeepSeek-R1-14B	21.4	18.1	32.3	69.0	12.0	26.6	64.5	13.8	28.5	65.9	17.2	35.8	83.9	7.8

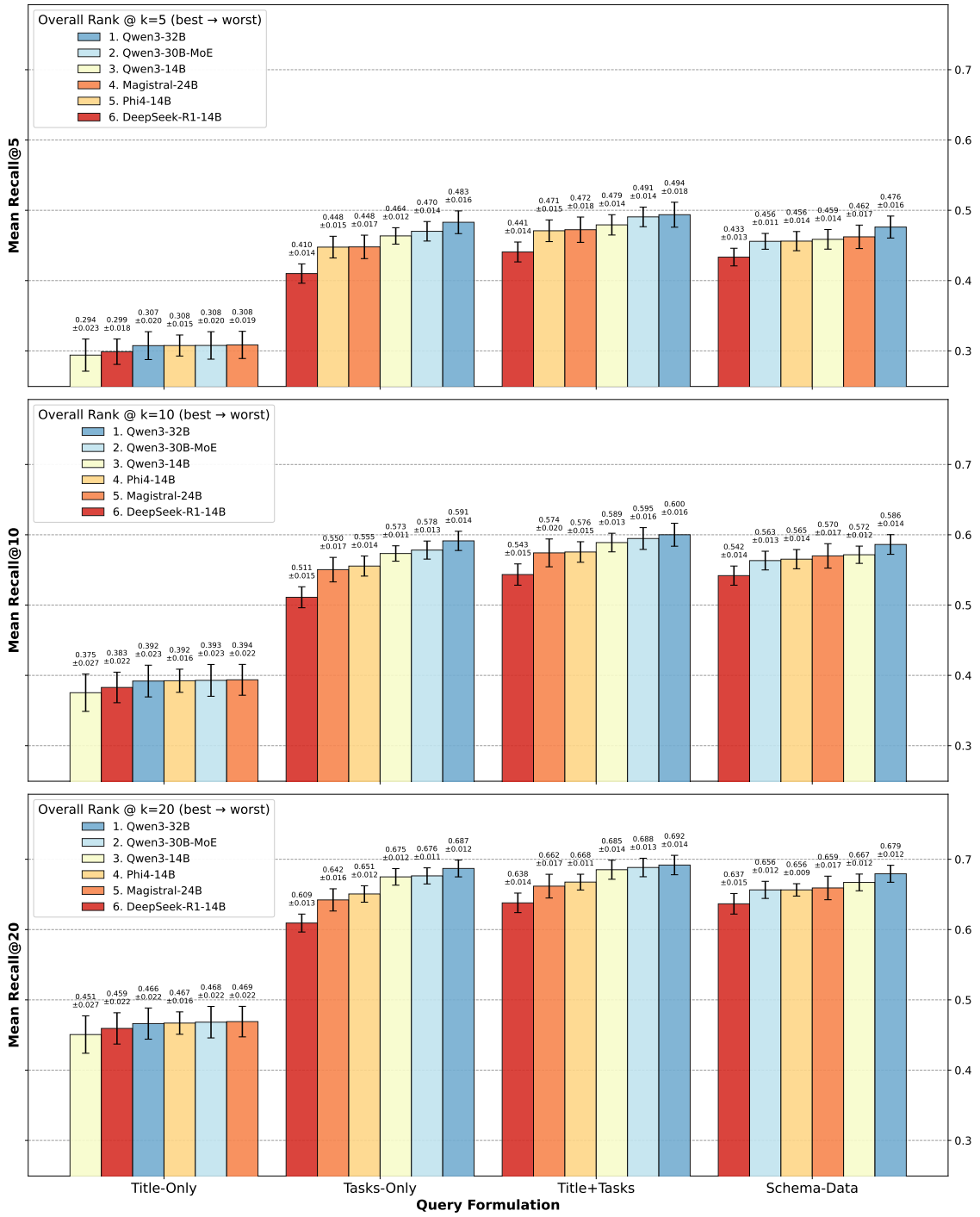


Figure 5.3: Impact of QR-LLMs on Recall Performance in QR-RAG. QR-LLMs exhibit narrow performance convergence (5.4pp spread at $k = 20$ with Title+Tasks) with Qwen3-32B achieving highest Recall (0.692) while Qwen3-30B-MoE delivers 99.4% of peak performance. Hypothetical tasks provide the dominant retrieval signal (Tasks-Only: 98%, Title-Only: 68% recovery rates relative to Title+Tasks), and $k = 20$ captures substantial long-tail gains paralleling Naive-RAG patterns. The figure displays aggregated Recall@k results covering 72 experimental settings (6 QR-LLMs × 4 QFs × 3 retrieval depths), averaged across nine embedding models and four evaluation datasets. Panels correspond to retrieval depths ($k = 5, 10, 20$). Within each panel, bars are grouped by QFs and sorted by QR-LLM performance (worst to best, left to right). Numeric annotations above bars represent the mean ± standard error of the mean (SEM) to quantify uncertainty across aggregates. Legends rank QR-LLMs by overall performance at each k-level (best to worst, top to bottom). Color coding is consistent across all panels for direct comparison.

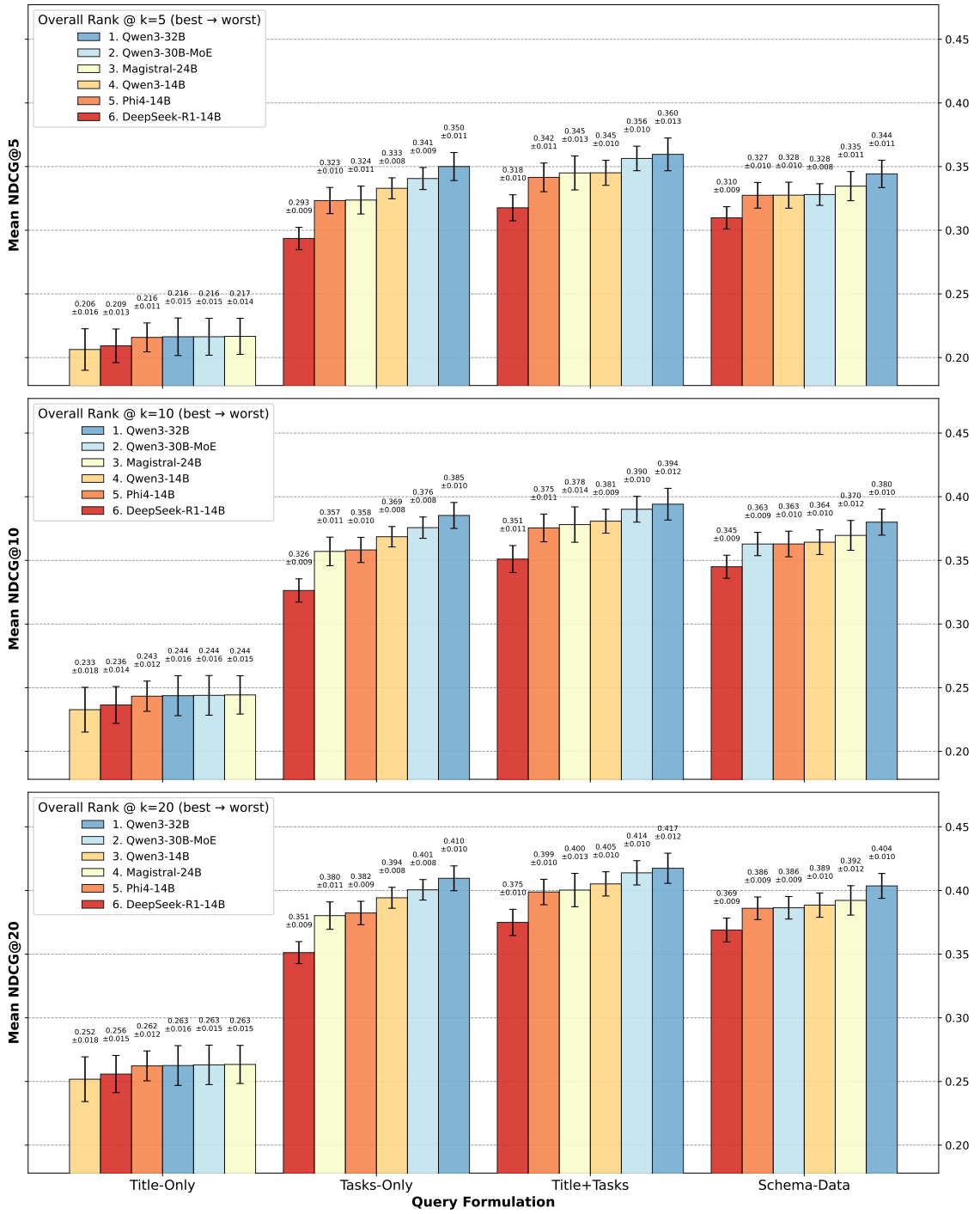


Figure 5.4: Impact of QR-LLMs on NDCG Performance in QR-RAG. NDCG metrics confirm the Recall patterns at $k = 20$ with Title+Tasks: narrow QR-LLM performance convergence with Qwen3-32B achieving highest ranking quality while Qwen3-30B-MoE delivers near-peak performance, hypothetical tasks providing the dominant retrieval signal, and $k = 20$ long-tail gains paralleling Naive-RAG. The figure displays aggregated NDCG@k results covering 72 experimental settings (6 QR-LLMs $\times$ 4 QFs $\times$ 3 retrieval depths), averaged across nine embedding models and four evaluation datasets. Panels correspond to retrieval depths ($k = 5, 10, 20$). Within each panel, bars are grouped by QFs and sorted by QR-LLM performance (worst to best, left to right). Numeric annotations above bars represent the mean $\pm$ standard error of the mean (SEM) to quantify uncertainty across aggregates. Legends rank QR-LLMs by overall performance at each k-level (best to worst, top to bottom). Color coding is consistent across all panels for direct comparison.

5.3.1 Qwen3-30B-MoE Bypasses the Quality–Efficiency Trade-off

We anticipated a quality–efficiency trade-off when using LLMs with higher parameter counts (from 14B to 32B) as query rewriters. Larger models should deliver superior retrieval quality through greater representational capacity by rewriting terse or ambiguous German free-text survey responses to better align with formal KldB 2010 descriptions [Jagerman et al., 2023; Koo et al., 2024; Ma et al., 2023], but they should also incur increased execution latency during query rewriting. This expectation reflects the conventional scaling paradigm in dense language models, where performance improvements come at direct computational cost [Zhang et al., 2025a].

Our measurements challenged this conventional trade-off (Figures 5.3 and 5.4, Table 5.6). Retrieval performance across QR-LLMs fell within a narrow range per query formulation, whereas execution times diverged substantially. At $k = 20$ using the Title+Tasks formulation, Qwen3-32B achieved the highest Recall at 0.692, while Qwen3-30B-MoE followed at 0.688—a negligible 0.4pp deficit representing 99.4% of peak performance. Even the lowest-performing model (DeepSeek-R1-14B) reached 0.638, creating only a 5.4pp spread across all six models. This narrow band persisted across all retrieval depths and query formulations for both metrics, narrowing further under Title-Only where all models converged within 1.9pp. Execution times, however, spanned a 2.6× range: Qwen3-32B required 18.9 hours to complete query rewriting for our evaluation datasets (3,656 queries), whereas Qwen3-30B-MoE finished in 7.2 hours—delivering comparable retrieval quality in 38% of the time.

Qwen3-30B-MoE fundamentally contradicts the anticipated trade-off that characterizes dense models. We attribute this to the sparse activation mechanism in MoE architectures [Yang et al., 2025; Zhang et al., 2025a]: only a subset of expert parameters activates per token, minimizing computational overhead while preserving the knowledge and text generation capabilities required for high-quality query rewriting. This property enables efficient scaling of model capacity, allowing MoE models to bypass the quality–efficiency dilemma rather than balance it. The narrow performance range further suggests that the capacity to bridge the gap between survey responses and taxonomic descriptions saturates beyond 14B parameters, making MoE efficiency gains particularly valuable where marginal quality improvements cannot justify substantial latency increases.

These findings answer the query-rewriter dimension of SQ-2, revealing three key insights. First, QR-RAG's retrieval performance across models with varying parameter counts (14B–32B) and architectures converged within a narrow 5.4pp range at $k = 20$ for the Title+Tasks formulation, indicating that parameter scale alone provides diminishing retrieval improvements beyond 14B. Second, architectural design decouples computational efficiency from retrieval quality, with MoE models eliminating the conventional trade-off. Third, optimal QR-LLM selection requires balancing these factors: Qwen3-30B-MoE achieves 99.4% of peak performance while executing 2.6× faster than the best-performing dense model.

Having answered the query-rewriter dimension of SQ-2, we apply these findings to our deployment strategy for SQ-3. We select Qwen3-30B-MoE as the optimal QR-LLM, where it serves dual roles as query rewriter for QR-RAG and as LLM-based

classifier for both RAG approaches. Beyond query rewriting, we employ this model for classification because its demonstrated abilities should transfer effectively to classification tasks requiring similar semantic bridging capabilities. For survey organizations with consumer-grade hardware and data-privacy requirements, these findings validate that local QR-RAG deployment remains computationally feasible: using concurrent batch prompting (Section 4.5.2), Qwen3-30B-MoE processes 3,656 queries in 7.2 hours ($\approx$ 7 seconds per query), enabling continuous survey coding without cloud infrastructure.

5.3.2 Hypothetical Tasks Provide the Dominant Retrieval Signal

We anticipated that high-density QFs would outperform the low-density Title-Only formulation. Each high-density formulation incorporates 10 LLM-generated hypothetical job-task sentences, whereas Title-Only relies solely on a rewritten job title in QR-RAG (Section 5.1.6). We further hypothesized that Schema-Data would offer superior discriminative power by integrating LLM-extracted occupational attributes (e.g., industry, specialization) from original survey data to improve query–document alignment with formal KldB 2010 descriptions.

Our measurements revealed a clear performance dichotomy among the query formulations that confirmed our first expectation while contradicting our second (Figures 5.3 and 5.4). At $k = 20$ for the optimal Qwen3-30B-MoE model, Title+Tasks achieved the highest Recall at 0.688, followed closely by Tasks-Only at 0.676 (1.2pp deficit) and Schema-Data at 0.656 (3.2pp deficit), while Title-Only trailed substantially at 0.468 (22.0pp deficit). Expressed as recovery rates relative to Title+Tasks, Tasks-Only achieved 98%, Schema-Data achieved 95%, and Title-Only achieved only 68%. This dichotomy remained stable across all retrieval depths and QR-LLMs for both metrics.

This performance pattern reveals that hypothetical tasks constitute the dominant retrieval signal in QR-RAG. The marginal 1.2pp gap between Title+Tasks and Tasks-Only indicates that once hypothetical tasks are present, the rewritten job title contributes minimal additional discriminative power. Given that original titles provide the dominant retrieval signal in Naive-RAG (Section 5.2.2), the combination of this finding with Title-Only's substantial performance deficit suggests that the title-rewriting process may lose or transform information critical for retrieval effectiveness. Regarding Schema-Data, its 3.2pp deficit relative to Title+Tasks contradicts our hypothesis that explicit occupational attributes would enhance discriminative power. We attribute this to signal dilution: because hypothetical task generation is conditioned on extracted occupational attributes, repeating these attributes side-by-side with the generated tasks as separate query fields in Schema-Data introduces noise that degrades query–document alignment during embedding rather than strengthening it.

These findings answer the context-density dimension of SQ-2, measured through QFs, revealing three key insights. First, hypothetical task generation constitutes the dominant mechanism of retrieval improvement in QR-RAG, with task-augmented formulations achieving 95–98% recovery rates relative to optimal performance. Second, title transformation alone appears to produce insufficient signal for effective retrieval, achieving only a 68% recovery rate and requiring task-level augmentation to approach optimal performance. Third, additional occupational fields provide no measurable

benefit beyond what is implicitly encoded in the generated tasks, as evidenced by Schema-Data's performance deficit.

Having answered the query-formulation dimension of SQ-2, we apply these findings to our experimental design for SQ-3. In classification experiments, we employ Title+Tasks as the most effective QF for both RAG approaches. In retrieval experiments, we retain all four QFs as an experimental dimension representing context density, enabling systematic evaluation of information-richness effects on system-level performance. For survey organizations implementing QR-RAG systems, these findings provide clear design guidance: query-rewriting implementations should prioritize hypothetical task generation, as this component provides the primary retrieval signal.

5.4 SQ-3: System-Level Retrieval Performance

Following the characterization of Naive-RAG's and QR-RAG's retrieval performance across individual embedding models and QR-LLMs in Sections 5.2 and 5.3, this section addresses SQ-3 by conducting a system-level comparison of both approaches. Our primary focus is to determine how different levels of information richness, represented by QFs with varying context density (Section 5.1.6) and DZHW evaluation datasets with varying input complexity (Section 5.1.1), influence the comparative retrieval effectiveness between both approaches across embedding-model performance tiers.

To ensure both RAG approaches operate at their highest achievable retrieval quality, we fix two key parameters based on prior findings: the optimal retrieval depth at $k = 20$ established in Section 5.2.3 and confirmed in Section 5.3, and the efficiency-optimal Qwen3-30B-MoE query rewriter identified in Section 5.3.1.

Our analysis covers 84 experimental settings generated by the cross-product of seven RAG-QF combinations, four evaluation datasets, and three embedding-model performance tiers (Table 5.7), stratifying results to examine whether QR effects vary systematically across EM capabilities. For each experimental setting, we report Recall@20 and NDCG@20 values aggregated over all EMs within the respective tier, controlling for individual model variability while revealing tier-specific patterns. Figures 5.5 and 5.6 display the retrieval performance across all experimental settings, organized by tier to facilitate cross-tier comparison of query-rewriting effectiveness.

For cross-system comparisons, we focus on the Title+Tasks multi-field formulation as the primary evidence to avoid the information asymmetry inherent in single-field formulations (Section 5.1.6). This asymmetry disappears in multi-field formulations, where both systems utilize the complete original input scope, enabling unbiased performance comparison. Single-field formulations are nevertheless included where they reveal cross-system differences in input-processing mechanisms and have been used to characterize the retrieval signal of individual input fields discussed below.

Our analysis proceeds along two complementary lines of investigation: (i) examining how QR effectiveness varies across embedding-model performance tiers (Section 5.4.1); and (ii) examining how both approaches respond to input-complexity variation (Section 5.4.2). Regarding retrieval signals, preliminary inspection confirms that the dominant signals remain stable in this cross-system context: job titles continue to

provide the strongest retrieval signal for Naive-RAG (Section 5.2.2), while hypothetical job tasks dominate for QR-RAG (Section 5.3.2). The system-level retrieval performance findings establish the foundation for evaluating classification effectiveness in Section 5.5, where we examine how retrieval performance translates into downstream classification accuracy.

Table 5.7: Embedding-Model Performance Tier Composition. The three performance tiers identified in Section 5.2.1 define discrete capability levels for German occupational retrieval. Top-tier models ($n = 2$) achieve 0.72–0.77 Recall@20 with Title+Tasks in Naive-RAG; mid-tier models ($n = 4$) achieve 0.51–0.62 (19–34% below peak); bottom-tier models ($n = 3$) achieve 0.15–0.37 (52–81% below peak). Results aggregate over models within each tier to examine tier-specific query-rewriting effects.

Performance Tier	Embedding Models
Top-Tier ($n = 2$)	Qwen3-Embedding, Snowflake-Arctic-Embed2
Mid-Tier ($n = 4$)	Multilingual-E5-Large, BGE-M3, Granite-Embedding, Jina-v2-Base-DE
Bottom-Tier ($n = 3$)	MxBai-Embed-Large, Paraphrase-Multilingual, EmbeddingGemma

5.4.1 LLM-Based QR Compensates for Weak Embedding Models

Our prior analyses established a 3-tier performance hierarchy among EMs in Naive-RAG, with only top-tier models achieving viable retrieval effectiveness (Section 5.2.1). We anticipated that LLM-based QR would provide uniform performance improvements across these embedding-model tiers, consistent with prior work demonstrating consistent gains from query-enhancement techniques across diverse retrieval scenarios [Jagerman et al., 2023; Koo et al., 2024; Ma et al., 2023].

Our measurements contradicted the expectation of uniform benefits, revealing systematic tier-dependent convergence in performance gaps between both approaches (Figures 5.5 and 5.6). Using the Title+Tasks formulation on the SJT dataset at $k = 20$, bottom-tier EMs exhibited large performance gaps: QR-RAG surpassed Naive-RAG by 33.5pp Recall (0.575 vs. 0.240, +140%) and 21.3pp NDCG (0.329 vs. 0.116, +185%). Mid-tier models showed substantially reduced gaps: 20.2pp Recall (0.710 vs. 0.508, +40%) and 15.8pp NDCG (0.424 vs. 0.266, +59%). Top-tier models exhibited minimal gaps: only 5.3pp Recall (0.749 vs. 0.696, +7.5%) and 6.2pp NDCG (0.465 vs. 0.403, +16%), representing an 84% gap reduction in Recall and 71% in NDCG from bottom to top tier. This convergence pattern held robustly across all four evaluation datasets for both metrics, with QR-RAG achieving universal superiority across all 12 tier-dataset combinations (mean advantages: 16.3pp Recall, 12.6pp NDCG).

The tier-dependent convergence reveals QR as a compensatory mechanism rather than a uniform improvement, with effectiveness inversely proportional to EM capability. Bottom-tier models, fundamentally limited in capturing German occupational terminology and handling data-quality issues (Section 5.2.1), derive substantial benefits from QR, as LLM-generated hypothetical tasks likely provide semantic scaffolding compensating for weak semantic understanding. Mid-tier models exhibit considerable benefits, with QR filling comprehension gaps in bridging the semantic gap. Top-tier models, already proficient at semantic mapping, show substantially smaller coverage

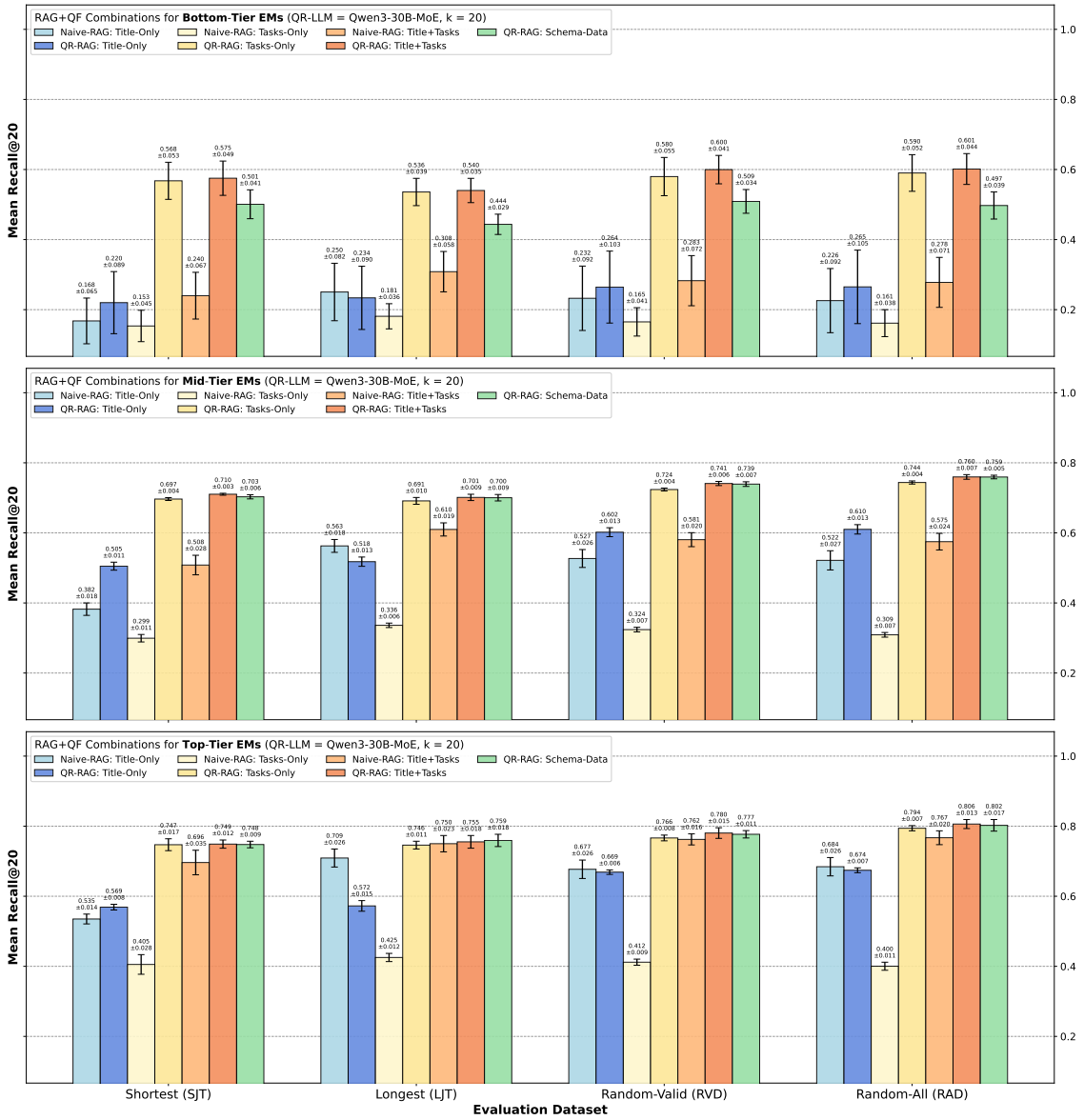


Figure 5.5: RAG-QF Comparison Across Embedding Tiers: Recall@20. Using the Title+Tasks formulation, QR-RAG consistently outperforms Naive-RAG across all tiers and datasets, with performance gaps decreasing systematically as embedding-model capability increases (84% gap reduction from bottom to top tier on SJT). Naive-RAG exhibits systematic sensitivity to input complexity (longer job titles in LJT improve performance across all tiers), while QR-RAG demonstrates near-complete insensitivity to input-quality variation. The figure displays aggregated Recall results at optimal $k = 20$ with Qwen3-30B-MoE as efficiency-optimal QR-LLM, covering 84 experimental settings (7 RAG-QF combinations $\times$ 4 datasets $\times$ 3 EM tiers), with values averaged within each performance tier. Panels correspond to embedding-model tiers (bottom-tier, mid-tier, top-tier, from top to bottom). Within each panel, bars are grouped by evaluation datasets, with RAG-QF combinations arranged as paired comparisons (Naive-RAG vs. QR-RAG per query formulation). Numeric annotations above bars represent the mean $\pm$ standard error of the mean (SEM) to quantify uncertainty across embedding models within the tier. Legends identify the specific RAG-QF combinations and indicate the tier scope for each panel. Color coding follows a QF-grouped scheme with light-dark pairs distinguishing Naive-RAG from QR-RAG within each formulation.

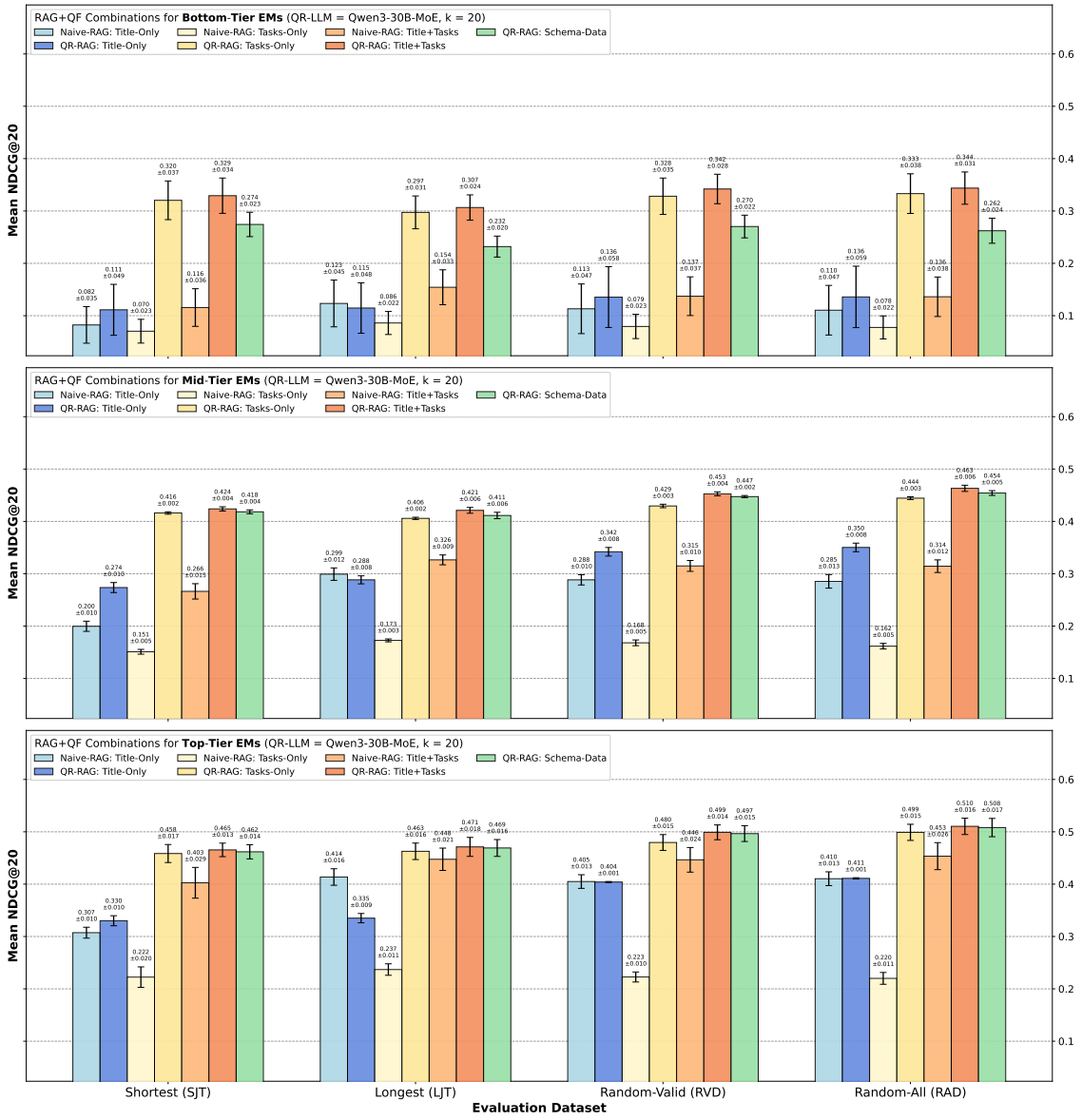


Figure 5.6: RAG-QF Comparison Across Embedding Tiers: NDCG@20. Using the Title+Tasks formulation, QR-RAG consistently outperforms Naive-RAG across all tiers and datasets, with performance gaps decreasing systematically as embedding-model capability increases (71% gap reduction from bottom to top tier on SJT). The weaker convergence compared to Recall (84%) indicates ranking quality benefits persist more strongly at top tier. Naive-RAG exhibits systematic sensitivity to input complexity (longer job titles in LJT improve performance across all tiers), while QR-RAG demonstrates near-complete insensitivity to input-quality variation. The figure displays aggregated NDCG results at optimal $k = 20$ with Qwen3-30B-MoE as efficiency-optimal QR-LLM, covering 84 experimental settings (7 RAG-QF combinations $\times$ 4 datasets $\times$ 3 EM tiers), with values averaged within each performance tier. Panels correspond to embedding-model tiers (bottom-tier, mid-tier, top-tier, from top to bottom). Within each panel, bars are grouped by evaluation datasets, with RAG-QF combinations arranged as paired comparisons (Naive-RAG vs. QR-RAG per query formulation). Numeric annotations above bars represent the mean $\pm$ standard error of the mean (SEM) to quantify uncertainty across embedding models within the tier. Legends identify the specific RAG-QF combinations and indicate the tier scope for each panel. Color coding follows a QF-grouped scheme with light-dark pairs distinguishing Naive-RAG from QR-RAG within each formulation.

gains as their strong representations already capture the relevant semantic space. However, the weaker convergence in NDCG versus Recall (71% vs. 84% gap reduction) reveals that ranking quality benefits persist even for top-tier models: hypothetical task generation continues to provide relevance signals improving document ordering even when coverage gains plateau. This compensatory pattern aligns with recent findings on LLM-based QR [Jagerman et al., 2023; Ma et al., 2023], though prior work did not examine interaction effects with retriever capability—our tier-stratified analysis reveals that QR benefits concentrate disproportionately among weaker retrievers.

These findings address part (i) of SQ-3 through tier-stratified analysis of retrieval performance comparison across information-richness conditions, revealing three key insights. First, LLM-based QR effectiveness exhibits strong inverse correlation with EM capability (84% Recall gap reduction, 71% NDCG gap reduction from bottom to top tier on SJT), contradicting the expectation of uniform improvements and establishing QR as a compensatory mechanism disproportionately benefiting weaker EMs. Second, QR benefits manifest differently for coverage versus ranking: top-tier models show smaller coverage gains (0.5–5.3pp Recall, 0.7–7.5% relative) but retain larger ranking improvements (2.3–6.2pp NDCG, 5.1–16% relative), indicating hypothetical task generation enhances relevance signals even when retrieval coverage plateaus. Third, QR-RAG achieves universal superiority across all 12 tier-dataset combinations (mean: 16.3pp Recall, 12.6pp NDCG; range: 0.5–33.5pp Recall), confirming QR provides measurable benefits across the full spectrum of EM capabilities and information-richness conditions, though with tier-dependent magnitudes.

Having addressed part (i) of SQ-3 through tier-stratified analysis, we observe that ranking quality benefits persist for top-tier models (2.3–6.2pp NDCG improvements) despite minimal coverage gains. This finding motivates our classification experiments in Section 5.5, evaluating whether these ranking improvements translate into classification performance gains for part (ii) of SQ-3. For practitioners, these findings provide tier-specific deployment guidance. Organizations with top-tier EMs and limited budgets may deploy Naive-RAG for coverage-focused applications (achieving 0.767 Recall without QR overhead, Table 5.6) but should adopt QR-RAG when ranking precision matters for downstream classifiers or user-review workflows. Organizations using mid-tier or bottom-tier EMs should prioritize QR-RAG, as the computational overhead (+7.2 hours for 3,656 queries with Qwen3-30B-MoE) is justified by substantial coverage gains required for viable performance (20.2–33.5pp Recall improvements).

5.4.2 LLM-Based QR Neutralizes Input-Complexity Effects

Our evaluation datasets operationalize the input-complexity dimension of information richness through job-title length as a proxy (Section 5.1.1), with SJT and LJT representing the shortest and longest job titles, respectively, alongside correlated differences in task-field completion (Table 3.9). We anticipated that higher input complexity would improve retrieval performance for both RAG approaches, as longer, more detailed job descriptions provide richer textual content for semantic matching.

Our measurements revealed a striking asymmetry in how both approaches respond to input complexity between SJT and LJT datasets (Figures 5.5 and 5.6). Using the Title+Tasks formulation at $k = 20$, Naive-RAG exhibited consistent performance gains

from longer inputs across all embedding-model tiers: bottom tier showed 6.8pp Recall advantage for LJT (0.308 vs. 0.240, +28%); mid tier maintained 10.2pp advantage (0.610 vs. 0.508, +20%); top tier retained 5.4pp advantage (0.750 vs. 0.696, +8%). In contrast, QR-RAG demonstrated near-complete insensitivity to input complexity: bottom tier showed reversed pattern with SJT outperforming LJT by 3.5pp Recall (0.575 vs. 0.540, -6%); mid tier equalized with only 0.9pp difference (0.710 vs. 0.701); top tier showed minimal 0.6pp LJT advantage (0.755 vs. 0.749). This asymmetric response extended to other query formulations: Title-Only and Tasks-Only confirmed Naive-RAG consistently benefits from longer inputs (+2.0 to +18.1pp Recall), with substantially larger differences for Title-Only (+8.3 to +18.1pp, the dominant retrieval signal) than Tasks-Only (+2.0 to +3.7pp), while QR-RAG showed substantially reduced sensitivity across formulations (-3.5 to +1.4pp Recall). NDCG exhibited the same asymmetric pattern across all formulations and tiers.

The asymmetric response reveals LLM-based QR as a normalizing mechanism that neutralizes input-complexity effects. In Naive-RAG, EMs directly process original input text, making retrieval quality sensitive to descriptive richness: longer, more detailed job descriptions provide more lexical and semantic signals for the EM to capture, resulting in more discriminative query representations. In QR-RAG, the LLM-based QR pipeline transforms all inputs into standardized, task-enriched representations of consistent structure and length (10 hypothetical task sentences plus transformed title). This transformation decouples retrieval performance from original input descriptiveness: the QR-LLM (Qwen3-30B-MoE) extracts occupational semantics from even minimal inputs and generates normalized query representations achieving comparable retrieval effectiveness regardless of source complexity.

These findings address part (i) of SQ-3 through input-complexity analysis of retrieval performance comparison across information-richness conditions, revealing two key insights. First, Naive-RAG exhibits systematic input-complexity sensitivity, with longer inputs improving performance across all tiers (5.4–10.2pp Recall advantages for LJT vs. SJT, 8–28% relative improvements) and formulations (Title-Only: +8.3 to +18.1pp; Tasks-Only: +2.0 to +3.7pp), confirming that retrieval effectiveness scales with input descriptiveness when EMs directly process original text. Second, QR-RAG demonstrates near-complete insensitivity to input complexity, with performance differences across datasets reduced to near-zero ranges (-3.5 to +1.4pp Recall, -6% to +1% relative), establishing QR as a normalizing mechanism that standardizes inputs regardless of source quality and effectively decouples retrieval performance from original input descriptiveness.

Having addressed part (i) of SQ-3 through input-complexity analysis, these findings establish QR as a normalizing mechanism for input-quality variation, providing complementary context for classification experiments (Section 5.5) where input standardization effects may influence classification performance. For practitioners, these findings provide deployment guidance focused on input quality. Organizations processing survey data with inconsistent input quality or limited preprocessing resources should prioritize QR-RAG, as it achieves stable performance regardless of input descriptiveness (-3.5 to +1.4pp variation) compared to Naive-RAG's substantial sensitivity (5.4–10.2pp variation). Organizations with high-quality, detailed input data and top-tier EMs may achieve adequate performance with Naive-RAG (0.750 Recall

on LJT), though QR-RAG's normalization benefits (+0.6pp on LJT, +7.9pp on SJT) suggest adopting QR provides insurance against input-quality degradation in operational settings. The computational overhead (+7.2 hours, Table 5.6) must be weighed against the operational value of robustness to input-quality variation.

5.5 SQ-3: System-Level Classification Performance

Having compared the system-level retrieval performance of Naive-RAG and QR-RAG with a focus on how different levels of information richness influence retrieval effectiveness across EM performance tiers (Section 5.4), this section addresses SQ-3 by conducting a system-level comparison of classification performance. Our focus is to determine how effectively high retrieval performance translates into downstream classification accuracy, and to characterize classification effectiveness across the five KldB hierarchy levels (1-digit to 5-digit codes).

To ensure both approaches operate at their highest achievable retrieval performance, we adopt the optimal configuration from system-level retrieval experiments: retrieval depth $k = 20$ and Qwen3-Embedding as the embedding model, with Qwen3-30B-MoE performing dual roles as the efficiency-optimal QR-LLM for QR-RAG and as the LLM-based classifier for both approaches. However, classification experiments require adapted execution parameters due to substantially larger prompt sizes. The classifier operates with a 19k-token context window (versus 5.4k for QR) and four concurrent requests, executing in standard mode without reasoning.

For classification experiments, we focus exclusively on the Title+Tasks formulation for two reasons: (i) it achieves the highest retrieval performance for both approaches (Sections 5.2.2 and 5.3.2, Figures 5.5 and 5.6), representing the practically most relevant configuration; and (ii) unlike single-field QFs that exhibit information asymmetry (Section 5.1.6), Title+Tasks enables unbiased cross-system comparison as both systems utilize the complete original input scope. Schema-Data is excluded as its structural differences from Title+Tasks would complicate direct performance attribution.

Our analysis covers 20 experimental settings generated by the cross-product of five KldB hierarchy levels (Table 1.1) and four DZHW evaluation datasets (Section 5.1.1). For each setting, we report Accuracy and Cohen's Kappa at $k = 20$, measuring both unconditional and conditional classification performance. Conditional metrics isolate classifier effectiveness from retrieval limitations, enabling fair evaluation of how well the LLM-based classifier converts successfully retrieved candidates into correct classifications. Figures 5.7 and 5.8 display classification performance across all experimental settings, visualizing both unconditional and conditional metrics to separate retrieval penalties from classification effectiveness.

Our analysis proceeds along four complementary lines of investigation: (i) characterizing performance-degradation patterns across hierarchy levels to identify practical viability thresholds for deployment (Section 5.5.1); (ii) examining whether data-quality differences between original and rewritten queries provided in classification prompts translate into performance differences (Section 5.5.2); (iii) evaluating the classifier's ability to convert successfully retrieved candidates into correct classifications,

determining whether retrieval or classification represents the primary performance bottleneck (Section 5.5.3); and (iv) contextualizing unsupervised performance against supervised fine-tuning baselines to establish hierarchy-specific deployment thresholds where training-data costs can be eliminated (Section 5.5.4).

5.5.1 3-to-4-Digit Transition Reveals Performance Knee Point

We first examine how classification performance degrades across the five hierarchy levels to identify practical viability thresholds for deployment. We anticipated monotonic performance degradation at deeper hierarchy levels. Each level introduces additional classification choices as the KldB 2010 expands dramatically: 10 occupational areas (1-digit) → 37 main groups (2-digits) → 144 groups (3-digits) → 702 subgroups (4-digits) → 1,300 types (5-digits). Each transition requires the LLM-based classifier to make increasingly fine-grained semantic distinctions, with deeper levels demanding more detailed occupational knowledge [Schierholz and Schonlau, 2020].

Our measurements confirmed monotonic degradation while revealing a sharp performance knee point at the 3-to-4-digit transition across all experimental settings (Figures 5.7 and 5.8). Using Naive-RAG on SJT, accuracy dropped 20.2pp from 3-digit (0.619) to 4-digit (0.417), while Cohen's Kappa declined 19.9pp (0.615 to 0.416). This 3→4 gap consistently exceeded all other transitional gaps (7.6–15.8pp accuracy), establishing a knee point where degradation rate sharply accelerates. The pattern spanned all datasets and both approaches: 18.4–20.2pp accuracy gaps (Naive-RAG), 17.3–19.4pp gaps (QR-RAG), and 17.0–19.9pp Kappa declines. Notably, the 4→5 transition produced smaller declines (13.0–16.1pp accuracy; 12.9–16.0pp Kappa) despite adding 598 classes—comparable to the 558-class expansion at 3→4—indicating the knee point represents a qualitative shift beyond simple class expansion.

The 3→4 knee point likely arises from dramatic class expansion intersecting with semantic granularity requirements. Upper levels maintain coarse-grained distinctions between broad occupational domains [Paulus and Matthes, 2013], while the 3→4 transition requires fine-grained differentiation at the subgroup level where distinctions involve subtle task variations. Safikhani et al. [2023] observed comparable degradation using fine-tuned PLMs on similar DZHW data, though their supervised approach achieved smaller gaps (≈ 8pp) than our unsupervised system's 17–20pp declines. The 4→5 transition shows attenuated loss despite comparable class expansion because the 5-digit level encodes requirement levels rather than occupational specialization [Paulus and Matthes, 2013], which the classifier can likely infer more reliably from educational qualifications and task complexity indicators within job descriptions.

These findings address part (ii) of SQ-3 regarding classification performance across KldB hierarchy levels, revealing three key insights. First, classification exhibits monotonic degradation with a sharp knee point at the 3→4 transition, where performance drops of 17–20pp consistently exceed other transitions, establishing this boundary as a qualitative threshold. Second, the 4→5 transition shows attenuated loss despite comparable class expansion, suggesting requirement-level classification enables more reliable inference than specialization-level distinctions. Third, both approaches exhibit nearly identical knee point patterns (mean 3→4 gap: 19.3pp vs. 18.4pp), confirming approach equivalence (Section 5.5.2) and establishing the knee point as a fundamental classification challenge rather than a retrieval-specific limitation.

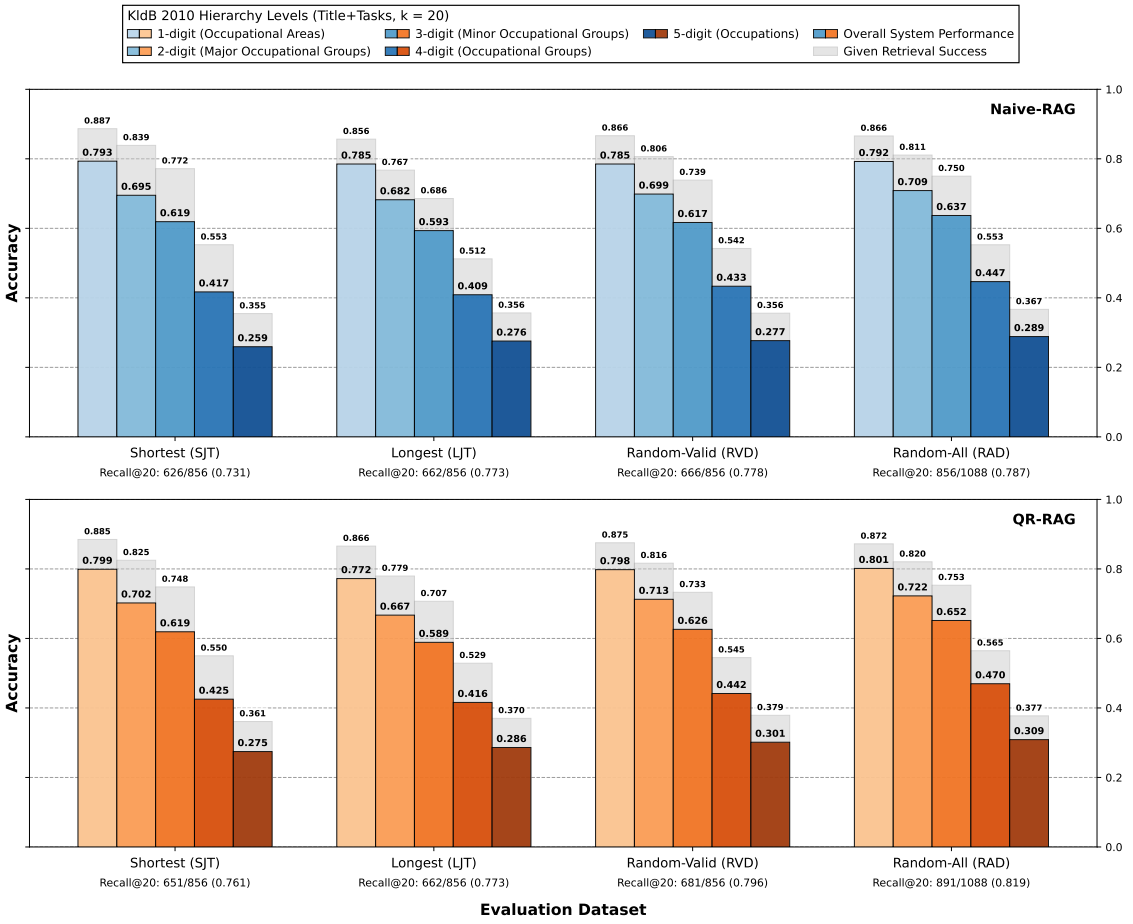


Figure 5.7: Accuracy Performance Across the KldB 2010 Hierarchy. Both RAG approaches exhibit minimal accuracy differences (-1.5 to +2.5pp) across all hierarchy levels, with a sharp performance decline at the 3-to-4-digit transition ($\approx$ 17–20pp) representing a knee point where degradation rate sharply accelerates beyond simple class expansion effects. Conditional accuracy at the 5-digit level (0.355–0.379) reveals that only 43–52% of successfully retrieved occupations are correctly classified, establishing classification rather than retrieval as the primary performance bottleneck—a limitation universal to both approaches. The figure compares Naive-RAG and QR-RAG classification accuracy across five KldB 2010 hierarchy levels at $k = 20$ using Title+Tasks formulation with Qwen3-Embedding and Qwen3-30B-MoE as classifier. Each panel presents one RAG approach. Bars group four evaluation datasets, each showing five hierarchy levels from 1-digit (Occupational Areas, lightest) to 5-digit (Occupational Types, darkest). X-axis labels show dataset-specific Recall@20 values. 2-layer bars separate unconditional accuracy (solid colored) from conditional accuracy given retrieval success (grey background), with gaps indicating retrieval penalties. Bold values above colored bars show unconditional accuracy; values above grey bars show conditional accuracy. Split-colored legend boxes indicate hierarchy levels: blue (left) for Naive-RAG, orange (right) for QR-RAG.

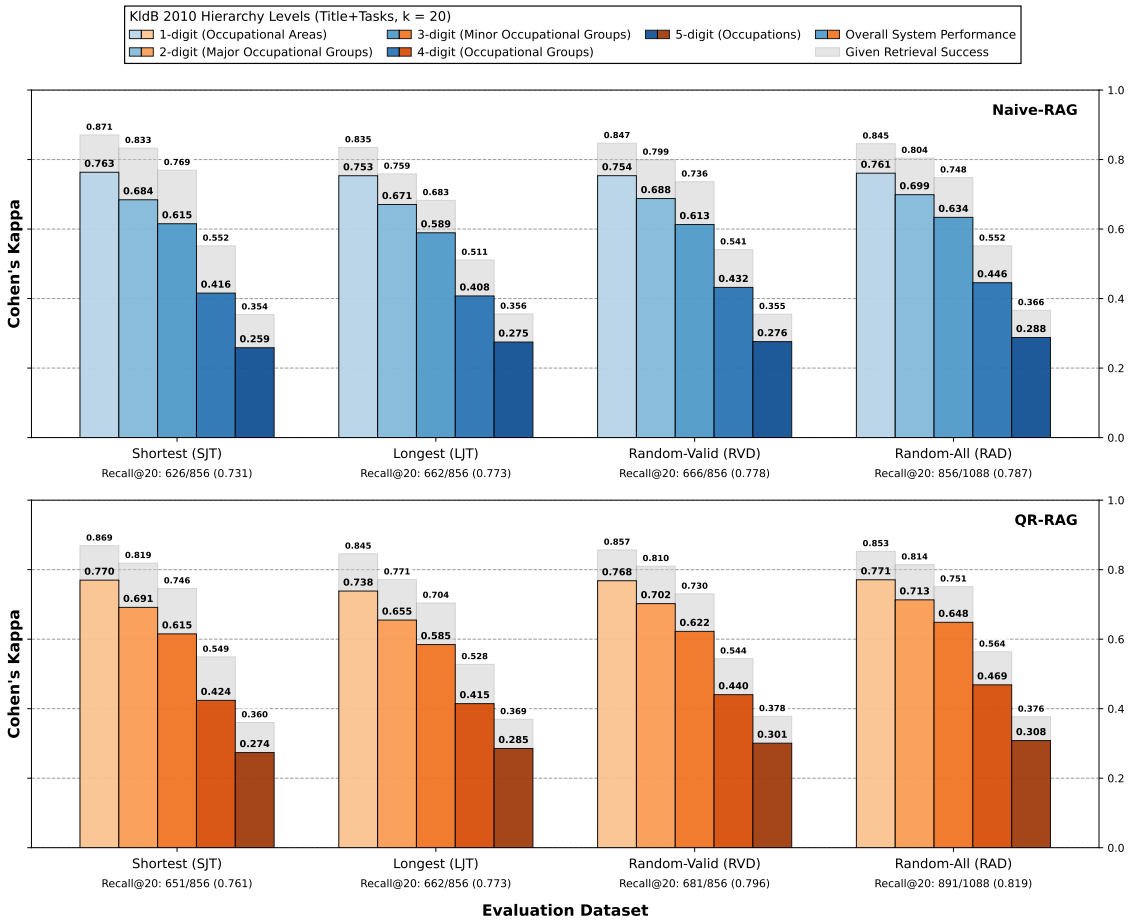


Figure 5.8: Cohen's Kappa Performance Across the KldB 2010 Hierarchy. Both RAG approaches exhibit minimal Cohen's Kappa differences (-1.5 to +2.4pp) across all hierarchy levels, indicating the LLM-based classifier neutralizes query-quality variations between approaches. The pattern shows a sharp decline at the 3-to-4-digit transition ($\approx$ 17–20pp) consistent with the accuracy knee point, representing a qualitative shift in classification difficulty where fine-grained semantic distinctions challenge the classifier's discrimination ability—a fundamental limitation affecting both approaches equally. Unsupervised performance matches supervised GPT-3 baselines at coarse levels (1–3 digits). The figure compares Naive-RAG and QR-RAG Cohen's Kappa across five KldB 2010 hierarchy levels at $k = 20$ using Title+Tasks formulation with Qwen3-Embedding and Qwen3-30B-MoE as classifier. Each panel presents one RAG approach. Bars group four evaluation datasets, each showing five hierarchy levels from 1-digit (Occupational Areas, lightest) to 5-digit (Occupational Types, darkest). X-axis labels show dataset-specific Recall@20 values. 2-layer bars separate unconditional Kappa (solid colored) from conditional Kappa given retrieval success (grey background), with gaps indicating retrieval penalties. Bold values above colored bars show unconditional Kappa; values above grey bars show conditional Kappa. Split-colored legend boxes indicate hierarchy levels: blue (left) for Naive-RAG, orange (right) for QR-RAG.

Having identified the 3→4 transition as a performance knee point, these findings establish level-specific deployment guidance for unsupervised occupation-coding systems. The 3-digit level may represent a practical viability threshold, with both approaches achieving 0.585–0.652 for accuracy and Cohen's Kappa—performance potentially acceptable for semi-automated coding workflows. Organizations requiring only broad categorization (1–3 digits) can deploy current unsupervised RAG architectures with acceptable performance, eliminating labeled training data and model-training costs. Organizations requiring fine-grained classification (4–5 digits) should recognize that current architectures likely require substantial improvements—such as hierarchical decomposition or reranking techniques—before achieving deployment-viable performance. For researchers, the universal knee point pattern suggests architectural innovations should target classifier improvements at the 3→4 transition rather than retrieval enhancements.

5.5.2 LLM-Based Classifier Neutralizes Query-Quality Effects

Having established performance patterns across hierarchy levels, we now examine whether the two RAG approaches differ in classification performance. Both approaches achieved similar recall performance under optimal settings (Section 5.4). While this recall similarity suggested similar classification performance might follow, we remained uncertain whether differences in data quality between original job descriptions (Naive-RAG) and query-rewritten versions (QR-RAG) provided to the LLM-based classifier alongside retrieved candidates would affect classification outcomes.

Our measurements resolved this uncertainty, revealing minimal classification differences between both approaches across all hierarchy levels and evaluation datasets (Figures 5.7 and 5.8). Using optimal settings (Qwen3-Embedding at $k = 20$ for Title+Tasks), accuracy differences ranged only from -1.5pp to +2.5pp across all experimental conditions, with Cohen's Kappa differences ranging from -1.5pp to +2.4pp. These minimal differences indicate approach choice has negligible impact on classification outcomes. To contextualize, observed differences are an order of magnitude smaller than the 3→4 transition performance drops (17–20pp, Section 5.5.1), confirming hierarchy-level complexity dominates over approach-specific variations.

These observations resolve our uncertainty and suggest the LLM-based classifier exhibits robustness to variation in classification prompts between approaches—both in which specific documents were retrieved despite similar overall recall, and in the input job descriptions provided (original vs. query-rewritten). QR-RAG enriches inputs with hypothetical job tasks, a quality advantage that previously improved retrieval for weaker embedding models (Section 5.4.1). Despite this advantage, the classifier (Qwen3-30B-MoE—the same model used for query rewriting) extracts occupational semantics from both variants with equivalent effectiveness, implicitly compensating for input-quality variations without requiring the explicit LLM prompts used in the QR stage. Under evaluated settings, input-data quality does not limit classification performance.

These findings address part (ii) of SQ-3 by establishing that approach choice between Naive-RAG and QR-RAG has minimal impact on classification outcomes under

optimal settings, resolving initial uncertainty about whether data-quality differences would affect classification despite similar retrieval. Performance differences are negligible (-1.5 to +2.5pp), confirming functionally equivalent outcomes. This equivalence indicates the LLM-based classifier exhibits robustness to data-quality variations in input job descriptions, effectively neutralizing QR-RAG's query-enrichment advantage that proved beneficial at the retrieval stage with weaker embedding models. Classification accuracy is primarily constrained by hierarchy-level complexity rather than approach-specific differences.

This performance equivalence establishes practical deployment guidance for RAG-based German occupation-coding systems. Organizations using optimal settings identified in this thesis can select Naive-RAG without the QR stage, eliminating LLM-inference costs while achieving classification performance equivalent to QR-RAG. Organizations constrained by weaker EMs should note that QR-RAG provides retrieval advantages compensating for model limitations (Section 5.4.1), potentially justifying QR overhead for such scenarios.

5.5.3 Low Conditional Accuracy Despite Strong Retrieval

Having established that both approaches achieved functionally equivalent classification performance despite high retrieval success (Section 5.5.2), we now examine whether retrieval or classification represents the primary performance bottleneck. We anticipated that the LLM-based classifier would achieve conversion rates of 60–80% at the 5-digit KldB level, based on the manageable candidate set size ($k = 20$). We operationalize this conversion efficiency using conditional accuracy: classification accuracy measured only on instances where the ground-truth occupation appears in the top-20 retrieved candidates, thereby isolating classifier decision-making from retrieval limitations.

Our measurements disconfirmed this expectation, revealing substantial gaps between retrieval success and classification accuracy across all hierarchy levels and evaluation datasets (Figures 5.7 and 5.8). At the 5-digit level, conditional accuracy reached only 0.355–0.379 across both approaches and all datasets, despite retrieval recall of 0.731–0.819 using the optimal Qwen3-Embedding model at $k = 20$. The classifier converted merely 43–52% of successfully retrieved ground-truth occupations into correct classifications, falling substantially short of our anticipated 60–80% conversion rate.

The low conversion efficiency reveals that successfully retrieving the correct occupational code among top-20 candidates does not guarantee correct classification. The classifier struggles to identify ground-truth occupations within candidate sets dominated by semantically similar descriptions. At the 5-digit granularity level, embedding models retrieve occupational descriptions that share substantial semantic overlap by design, creating severe discrimination challenges even when the correct occupation is present. This challenge aligns with the sharp performance decline at the 3-to-4-digit transition (Section 5.5.1), where nearly 5× class expansion introduces difficulties in distinguishing among hundreds of closely related occupational subgroups. Additionally, 1.86% of evaluation instances exhibit ground-truth ambiguity where identical job descriptions received different expert-assigned codes (Section 3.5.3),

rising to 3.86% in the SJT dataset, meaning classifiers may predict valid alternatives yet be scored as incorrect—suggesting these conditional accuracy metrics represent lower bounds on true classification capability.

These findings address part (ii) of SQ-3 by establishing classification as the primary performance bottleneck for RAG-based German occupation coding at the finest granularity level, revealing three key insights. First, classifier conversion efficiency at the 5-digit level reaches only 43–52%, falling substantially short of anticipated 60–80% conversion rates and indicating that retrieval success does not translate reliably into classification accuracy. Second, the substantial gap between retrieval success and conditional classification accuracy establishes classification rather than retrieval as the primary bottleneck, with the classifier unable to discriminate effectively among semantically similar candidates even when ground-truth occupations are successfully retrieved. Third, both Naive-RAG and QR-RAG exhibit identical conditional accuracy patterns across all datasets, confirming the bottleneck stems from fundamental classifier discrimination challenges at fine-grained taxonomic levels rather than approach-specific limitations.

These findings establish that future work should prioritize classification improvements over retrieval optimizations for both researchers and practitioners implementing RAG-based German occupation coding at the 5-digit level. Promising research directions include hierarchical classification strategies that leverage stronger 3-digit predictions as scaffolding for finer-grained distinctions, specialized LLM prompts leveraging reasoning capabilities (e.g., chain-of-thought prompting) that explicitly highlight discriminative features among semantically similar candidates, and reranking techniques that refine candidate ordering before final classification.

5.5.4 Unsupervised RAG Matches Supervised at Coarse Levels

Having established classification as the primary performance bottleneck at the 5-digit level (Section 5.5.3), we now contextualize our unsupervised RAG approaches against supervised fine-tuning baselines to assess the performance trade-off between methods requiring no labeled training data and those demanding extensive expert-coded examples. We anticipated that supervised fine-tuned models would substantially outperform unsupervised approaches across all hierarchy levels, given their access to thousands of labeled training instances and task-specific optimization. Safikhani et al. [2023] established supervised baselines on similar DZHW survey data, with fine-tuned BERT achieving Cohen's Kappa ranging from 0.846 at the 1-digit level to 0.622 at the 5-digit level, and fine-tuned GPT-3 achieving 0.760 to 0.516 across the same hierarchy (see their "Tables 6 and 7"), suggesting substantial unsupervised–supervised gaps across all levels.

Our measurements confirmed this performance gap at fine-grained levels while revealing an unexpected convergence at coarser hierarchy levels (Figures 5.7 and 5.8). At the 5-digit level, both unsupervised approaches achieved Cohen's Kappa ranging from 0.259–0.288 for Naive-RAG and 0.274–0.308 for QR-RAG across datasets, establishing performance gaps of 0.21–0.26 relative to supervised GPT-3 (0.516) and 0.31–0.36 relative to supervised BERT (0.622). These gaps reflect the fundamental trade-off: unsupervised approaches require no labeled training data and enable

immediate deployment, while supervised methods demand expensive expert-coded training examples with thousands of previously coded survey responses. However, these performance gaps diminish dramatically at coarser hierarchy levels. At the 1-digit level, QR-RAG achieved Cohen's Kappa of 0.738–0.771 across datasets, spanning the supervised GPT-3 performance of 0.760. At the 2-digit level, QR-RAG reached 0.655–0.713, matching supervised GPT-3 performance of 0.698. At the 3-digit level, QR-RAG achieved 0.585–0.648, matching supervised GPT-3 performance of 0.642. Fine-tuned BERT consistently outperformed all unsupervised approaches across all hierarchy levels by 0.08–0.36, establishing an upper performance bound that unsupervised methods did not reach.

This hierarchy-dependent convergence reveals that unsupervised RAG approaches possess sufficient semantic understanding for broad occupational categorization without requiring supervised fine-tuning. At coarser hierarchy levels (1–3 digits), the classification task involves distinguishing among broad occupational domains where semantic differences remain substantial [Paulus and Matthes, 2013], allowing LLM-based classifiers to leverage their general occupational knowledge effectively. The performance gap at finer levels (4–5 digits) arises from the need to discriminate among semantically similar subgroups (Section 5.5.1), where supervised models' task-specific training on expert-coded patterns provides advantages that unsupervised approaches cannot overcome without architectural innovations. The convergence at coarser levels indicates that the primary limitation of unsupervised RAG lies in fine-grained discrimination rather than broad occupational understanding.

These findings address part (ii) of SQ-3 by establishing the hierarchy-level conditions under which unsupervised RAG approaches achieve performance comparable to supervised fine-tuned models, revealing three key insights. First, unsupervised approaches incur substantial performance penalties (0.21–0.36 Cohen's Kappa deficit) relative to supervised baselines at the finest 5-digit level, confirming that task-specific fine-tuning provides measurable advantages for fine-grained classification. Second, this performance gap disappears at coarser hierarchy levels, with QR-RAG spanning supervised GPT-3 performance at the 1-digit level (0.738–0.771 vs. 0.760) and matching it at both the 2-digit level (0.655–0.713 vs. 0.698) and 3-digit level (0.585–0.648 vs. 0.642), establishing hierarchy-specific deployment thresholds. Third, both Naive-RAG and QR-RAG exhibit comparable performance at coarse levels (Section 5.5.2), confirming that the supervised-level performance represents a capability of unsupervised RAG architectures rather than QR-specific advantages.

These findings establish clear deployment guidance based on organizational granularity requirements and resource constraints. Organizations needing only broad occupational categorization (1–3 digit codes) can deploy unsupervised RAG approaches as viable alternatives to supervised methods, eliminating the substantial costs of generating thousands of expert-coded training examples while achieving performance equivalent to supervised fine-tuned models. This cost-benefit advantage proves particularly compelling for research institutions and survey organizations with limited labeled data or those requiring rapid deployment without model-training overhead. Organizations requiring fine-grained classification (4–5 digits) should recognize that current unsupervised architectures remain substantially below supervised baselines and likely require either supervised fine-tuning or architectural

improvements—such as hierarchical classification strategies or specialized prompting techniques—before achieving deployment-viable performance at these levels. For researchers, these findings suggest that unsupervised RAG architectures possess sufficient general occupational knowledge for coarse-grained tasks, directing future innovation efforts toward improving fine-grained discrimination capabilities rather than fundamental semantic understanding.

5.6 Summary

In this chapter we evaluated Naive-RAG and QR-RAG for unsupervised KldB 2010 occupation coding on German free-text survey data, addressing the three subquestions that together answer the main research question.

For Naive-RAG (SQ-1), embedding models separated into three discrete performance tiers rather than showing gradual differences, with only the top tier achieving sufficient retrieval coverage ($\geq$ 0.72 Recall@20) for viable deployment. This stratification suggests different model capabilities to bridge the semantic gap—spanning register and granularity mismatches—between colloquial survey language and formal KldB descriptions. Among query formulations, job titles—not task descriptions—proved to be the dominant retrieval signal, attributed to their mandatory coverage and conciseness. A retrieval depth of $k = 20$ was established as optimal, as the long-tail gains it delivers outweigh the marginal increase in computational load.

For QR-RAG (SQ-2), retrieval performance across all six QR-LLMs converged within a narrow range, indicating that the capacity to bridge the survey-to-taxonomy vocabulary gap saturates beyond 14B parameters. Qwen3-30B-MoE bypassed the conventional quality–efficiency trade-off entirely, matching the best dense model while executing 2.6× faster—attributed to its sparse activation mechanism. Hypothetical job-task generation emerged as the dominant retrieval mechanism in QR-RAG, shifting the primary signal from the original job title to LLM-generated task descriptions.

At the system level (SQ-3), LLM-based query rewriting acted as two complementary mechanisms at the retrieval level: it compensated for weak embedding models—providing the largest gains where retrieval coverage was lowest—and it neutralized input-complexity effects, transforming variable-quality survey inputs into standardized representations of consistent retrieval effectiveness. At the classification level, however, both approaches performed similarly across all KldB hierarchy levels, indicating that the LLM-based classifier neutralizes the query-quality advantage that query rewriting produces at the retrieval stage. Classification performance revealed a sharp knee point at the 3-to-4-digit transition, where accuracy and Cohen's Kappa dropped substantially beyond all other inter-level gaps, identifying the classifier—not the retriever—as the primary performance bottleneck. At coarse levels (1–3 digits), both approaches matched the fine-tuned GPT-3 baseline, establishing that label-free RAG is a viable alternative to supervised coding for broad occupational classification. The findings of this chapter directly inform the implications, limitations, and future-work directions discussed in Chapter 6.

6. Conclusion

Occupation coding underpins labor-market research and epidemiological studies. Expert coders manually assign occupation codes across diverse data sources, yet survey-based occupation coding presents distinctive challenges. Surveys typically collect occupational data as free-text responses to open-ended questions, creating unique coding difficulties. Expert coders require specialized training, achieve approximately 120 codes per working day, and exhibit inter-coder agreement ranging from 50–86% even among trained professionals.

The semantic gap between free-text survey responses and formal taxonomic definitions drives these difficulties. Register mismatches separate colloquial respondent language from standardized KldB terminology. Granularity mismatches separate position-oriented job descriptions from category-oriented classifications. Two characteristics of survey responses compound these difficulties. Terminological diversity creates substantial variation in how respondents describe identical occupational categories. Variable response density ranges from minimal generic titles to extensive descriptions including tasks, qualifications, hierarchical position, and industry context. These characteristics produce severe classification ambiguity. Resource constraints—personnel costs, processing time, and scale limitations—further amplify these challenges, making consistent coding quality difficult across large surveys and longitudinal studies.

Recent advances in RAG architectures and PLMs offer new automation opportunities by enabling semantic bridging between colloquial survey responses and standardized occupation taxonomies. However, their effectiveness for German survey-based occupation coding using the KldB 2010 taxonomy remained empirically uncharacterized. This absence created practical barriers for research institutions implementing occupation-coding workflows under data-privacy and computational constraints. This thesis addressed these gaps by establishing empirical baselines for unsupervised RAG-based occupation coding and systematically evaluating whether LLM-based query rewriting provided measurable performance improvements under realistic deployment conditions.

This chapter synthesizes our findings and situates them within broader theoretical and practical contexts. Section 6.1 articulates the research gaps motivating this investigation and specifies our complementary aims. Section 6.2 summarizes the six key contributions emerging from our evaluation of Naive-RAG and QR-RAG systems. Section 6.3 reviews the thesis structure and the progression from foundations through implementation to evaluation. Section 6.4 translates empirical findings into actionable guidance for practitioners deploying occupation-coding systems, architects designing RAG pipelines for hierarchical classification, and researchers evaluating classification systems with survey data. Section 6.5 addresses external and internal validity constraints affecting the generalizability and interpretation of our results. Section 6.6 proposes targeted improvements addressing identified performance bottlenecks. Finally, Section 6.7 explores broader research directions whose potential benefits remain empirically uncharacterized.

6.1 Research Gaps and Aims

Despite growing interest in automated occupation coding, no RAG-based baseline existed for KldB 2010 occupation coding under realistic computational constraints prior to this work. While RAG architectures had demonstrated success in question-answering and information-retrieval tasks, and recent work had applied RAG to occupation coding on international taxonomies [Achananuparp et al., 2025], their application to KldB 2010 occupation coding from German survey data remained unexplored. More critically, the relative merits of standard RAG versus query-rewriting-enhanced RAG for occupation coding lacked systematic evaluation. No prior work had isolated the performance contributions of individual system components—embedding models, LLM-based query rewriters, and classifiers—nor identified the conditions under which query rewriting provided meaningful advantages.

This thesis pursued two complementary aims. First, we established RAG-based occupation coding baselines for KldB 2010 by implementing and evaluating both standard RAG (Naive-RAG) and query-rewriting-enhanced RAG (QR-RAG) under controlled experimental conditions. Second, we systematically assessed the extent to which QR-RAG outperformed Naive-RAG in retrieval and classification performance, identifying the specific conditions—embedding-model capability tiers, information-richness dimensions (input complexity, context density), and KldB hierarchy levels—where performance improvements proved most pronounced. We addressed these aims through three complementary subquestions that isolated component-level effects by characterizing the retrieval performance of each approach separately, before conducting system-level comparisons that evaluated how both approaches performed under varying information-richness conditions and across hierarchy levels.

6.2 Main Results and Contributions

Our systematic evaluation of both approaches yielded six key contributions to RAG-based German occupation coding: (i) QR-RAG's universal retrieval superiority, (ii) two mechanisms explaining this advantage, (iii) no classification advantage despite retrieval gains, (iv) identical bottlenecks at fine-grained levels, (v) a universal discrimination knee point at the 3-to-4-digit transition, and (vi) hierarchy-dependent

convergence with supervised baselines. These findings are organized below by how they build upon one another through the retrieval and classification pipeline.

QR-RAG achieves universal retrieval superiority, improving performance by 16.3pp Recall and 12.6pp NDCG on average across all experimental conditions, with gains spanning 5.3pp to 33.5pp depending on deployment context. This provides the first systematic quantification of query-rewriting benefits for KldB 2010 occupation coding. Prior work applying RAG to international taxonomies had not employed LLM-based query rewriting through hypothetical task generation, leaving effectiveness for German occupation coding empirically uncharacterized.

Two mechanisms explain QR-RAG's retrieval advantages. First, query rewriting operates as a compensatory mechanism inversely correlated with embedding-model capability. Bottom-tier models gained 34pp Recall (140% improvement), mid-tier models 20pp (40%), and top-tier models only 5.3pp (7.5%), representing an 84% gap reduction from weakest to strongest models. This tier-stratified analysis is the first to characterize how QR benefits vary with EM capability in RAG-based German occupation coding, enabling weaker models to approach the performance of stronger retrievers. Second, query rewriting transforms the dominant retrieval signal from original job titles (achieving 88% of optimal performance in Naive-RAG) to LLM-generated hypothetical tasks (achieving 98% in QR-RAG). This signal transformation neutralizes input-complexity variation through standardized task generation. These mechanistic insights provide the first explanation of how and why query rewriting improves retrieval for occupation coding, extending beyond this domain to inform query-augmentation strategies for other heterogeneous or colloquial inputs such as patient-reported symptoms.

Classification performance shows no QR-RAG advantage despite retrieval gains. Under optimal settings where both approaches achieved near-equal retrieval performance, classification differences ranged only -1.5pp to +2.5pp in accuracy across all hierarchy levels and datasets. Given the absence of prior RAG-based baselines for KldB 2010 occupation coding, this stage-wise decomposition provides the first characterization of where performance losses occur within RAG pipelines—isolating classification rather than retrieval as the primary performance constraint.

Both approaches suffer identical classification bottlenecks at fine-grained levels. Only 43–52% of successfully retrieved ground-truth occupations were correctly classified at the 5-digit level. Performance degraded sharply at the 3-to-4-digit transition, with accuracy drops of 17–20pp affecting both approaches equally. While previous work on supervised fine-tuning observed performance degradation at this transition, our study extends these findings by establishing the 3-to-4-digit boundary as a universal discrimination knee point affecting unsupervised RAG approaches as well, with substantially larger gaps (17–20pp) than supervised methods (8pp).

Unsupervised RAG matches supervised baselines at coarse hierarchy levels. QR-RAG matched supervised GPT-3 performance at the 2-digit (0.655–0.713 vs. 0.698) and 3-digit levels (0.585–0.648 vs. 0.642), while incurring gaps of 0.21–0.26 at the 5-digit level. This provides the first systematic comparison between unsupervised RAG and supervised fine-tuned models for KldB 2010 occupation coding, establishing hierarchy-specific deployment thresholds where training-data costs can be eliminated.

The hierarchy-dependent convergence demonstrates that unsupervised RAG possesses sufficient general occupational knowledge for broad categorization, enabling organizations to deploy immediately at coarser granularity levels.

6.3 Thesis Organization

We organized the thesis to progress from contextual foundations through technical development to comprehensive empirical evaluation. Chapter 1 established the research context by examining the KldB 2010 taxonomy and characterizing the semantic gap in survey-based occupation coding. The chapter reviewed the evolution of automated coding approaches—from rule-based systems through supervised machine learning to emerging LLM-based methods—and identified technological opportunities with RAG architectures and multilingual PLMs. Building on this foundation, it defined the specific research problem: the lack of empirical characterization for unsupervised RAG-based German occupation coding under resource constraints. The chapter then articulated two research objectives addressing baseline establishment and query-rewriting effectiveness, formalized through one main research question and three subquestions that guided this investigation.

Chapter 2 established the theoretical and technical foundations required to follow the system design and evaluation decisions made throughout the thesis. The chapter traced the evolution from rule-based language processing to transformer-based PLMs, detailing tokenization, self-attention, feed-forward knowledge storage, and the encoder–decoder distinction that separates comprehension-oriented models from generative decoder models; it further explained how scaling these architectures into LLMs enables in-context learning and how quantization and MoE innovations make local deployment feasible under hardware constraints. Building on this, the chapter explained how text embedding models convert language into fixed-length geometric vectors and how the dual-encoder architecture decouples offline document indexing from online query encoding to enable scalable semantic retrieval. Finally, it motivated the RAG paradigm by characterizing the limits of purely parametric knowledge, described the indexing–retrieval–generation pipeline, and introduced LLM-based query rewriting—including hypothetical document generation—as the mechanism for bridging the register and granularity mismatches between informal survey responses and formal KldB taxonomy entries that lie at the core of this thesis.

Chapter 3 addressed the critical challenge of creating authentic evaluation data by developing a 3-dimensional DQAF that systematically classified data-quality issues along dimensions of origin, preprocessing feasibility, and LLM-addressability. A 6-step preprocessing pipeline was implemented to resolve syntactic irregularities while intentionally preserving the authentic linguistic challenges inherent in survey responses. This chapter further detailed the construction of four stratified evaluation datasets—LJT, SJT, RVD, and RAD—utilizing controlled sampling to operationalize input complexity via job-title length while ensuring taxonomic representativeness and computational feasibility for empirical evaluation.

Chapter 4 documented the design and implementation of both RAG-based occupation-coding systems. The chapter justified the selection of a vector-based RAG paradigm, contrasted the Naive-RAG and QR-RAG pipeline architectures, and explained

how data-privacy requirements and consumer-grade hardware constraints shaped implementation decisions. It detailed the construction of the shared knowledge base from all 1,300 KldB occupational descriptions, establishing task descriptions as the sole retrieval-relevant content field and indexing them into nine model-specific vector-store instances. The shared retrieval and classification stages were then specified, including cosine-similarity search over HNSW-indexed vectors and an LLM-based German-language classification prompt. Finally, the chapter detailed Naive-RAG's three direct query-formulation strategies and QR-RAG's three-stage query-rewriting pipeline—comprising content refinement, occupational attribute extraction, and hypothetical task generation—together with its concurrent batch-prompting architecture and four query-formulation strategies, including the Schema-Data formulation exclusive to QR-RAG that integrates structured occupational attributes alongside hypothetical job tasks to maximize query–document alignment.

Chapter 5 conducted the comprehensive evaluation addressing the three subquestions and, through their integration, the main research question of this thesis. We first characterized Naive-RAG's retrieval performance by isolating the effects of individual embedding models, then evaluated QR-RAG's retrieval effectiveness by isolating the effects of individual QR-LLMs. Using the optimal configurations identified through these component-level analyses, system-level comparisons were conducted to examine (i) how information richness moderated QR benefits for retrieval across three EM performance tiers and (ii) how effectively retrieval performance translated into classification performance across five KldB hierarchy levels. Collectively, these investigations answered the main research question by characterizing the extent and nature of performance differences between both approaches, identifying the conditions under which any improvements or trade-offs emerge, and providing actionable guidance for practitioners and researchers implementing unsupervised RAG-based occupation-coding systems.

6.4 Implications

Our evaluation generated empirical evidence supporting practical deployment decisions, architectural design choices, and methodological approaches for RAG-based classification systems. The systematic performance characterization across embedding-model tiers, hierarchy levels, and information-richness conditions enables practitioners to make informed deployment decisions under resource constraints. The mechanistic insights into query rewriting's compensatory and normalizing functions inform architectural choices for hierarchical classification tasks. The DQAF and stratified dataset-construction methodology provide replicable evaluation frameworks extending beyond occupation coding to survey-based classification more broadly. We organize implications along three dimensions: deployment guidance for practitioners implementing occupation-coding systems, architectural insights for RAG-system designers, and methodological guidance for researchers evaluating classification tasks with authentic survey data.

6.4.1 Deployment Guidance for Practitioners

Statistical agencies and research institutions implementing RAG-based occupation-coding systems face deployment decisions constrained by data-privacy requirements,

computational resources, and granularity requirements. Our findings inform these decisions along two dimensions: system-architecture selection and hierarchy-level deployment viability.

6.4.1.1 System-Architecture Selection

Embedding-model capability determines whether QR investment proves cost-effective. Deployments using weak or mid-tier models (e.g., due to license restrictions) benefit substantially from QR-RAG: retrieval improvements of 20–34pp Recall enable mid-tier models to approach top-tier Naive-RAG performance through the compensatory mechanism, justifying the computational overhead of approximately seven seconds per query. Conversely, deployments with top-tier EMs achieve comparable retrieval and classification performance using Naive-RAG, as marginal retrieval advantages provide insufficient classification improvements to justify QR overhead. Decision criterion: prioritize QR-RAG for weak and mid-tier models; evaluate whether marginal top-tier gains justify costs for resource-constrained contexts.

6.4.1.2 Hierarchy-Level Deployment Viability

Performance analysis across KldB levels establishes granularity-dependent deployment thresholds. Coarse-grained coding (1–3 digits) represents viable deployment contexts: unsupervised RAG approaches match supervised fine-tuned baselines while eliminating training-data costs. Institutions requiring only broad occupational categorization can therefore deploy immediately; architecture selection depends on embedding-model capability (Section 6.4.1.1). Fine-grained coding (4–5 digits) faces substantial classification bottlenecks: fewer than half of successfully retrieved occupations are correctly classified at the 5-digit level, with unsupervised approaches showing substantial performance deficits relative to supervised baselines. Institutions requiring fine-grained classification should consider supervised fine-tuning approaches or defer deployment pending architectural improvements addressing the discrimination bottleneck.

6.4.2 Architectural Insights for RAG-System Designers

Our evaluation generated three architectural insights for RAG systems addressing hierarchical classification tasks: performance-bottleneck localization, query-rewriting mechanisms, and hierarchy-specific discriminability limits.

6.4.2.1 Classification-Bottleneck Localization

Performance decomposition into retrieval and classification stages revealed classification as the primary constraint. At the 5-digit level, both approaches achieved strong retrieval coverage (Recall@20 ranging from 73–82% across datasets), yet conditional accuracy remained low, indicating fewer than half of successfully retrieved occupations were correctly classified. This establishes classification effectiveness, not retrieval coverage, as the critical determinant of overall system performance. Architectural improvements should therefore prioritize enhancing discrimination among semantically similar candidates over expanding retrieval depth or coverage.

6.4.2.2 Query-Rewriting Dual Mechanisms

Mechanistic analysis revealed that effective query rewriting operates through two complementary mechanisms. The compensatory mechanism exhibits inverse correlation with embedding-model capability: weak embedding models gained 140% relative improvement, while top-tier models showed only 7.5% relative improvement. This enables resource-constrained deployments to substitute preprocessing investment for model capacity. The normalizing mechanism eliminates input-complexity sensitivity by transforming variable inputs into standardized representations: retrieval dependence shifted from original job titles to LLM-generated hypothetical tasks, decoupling performance from input-quality variation. Together, these mechanisms democratize RAG retrieval performance across model tiers and input conditions.

6.4.2.3 Hierarchy-Specific Discriminability Limits

The 3-to-4-digit transition exhibited sharp performance degradation affecting both approaches identically, suggesting fundamental discriminability limits at fine taxonomic granularity independent of retrieval quality. This knee point establishes that retrieval improvements alone cannot overcome fine-grained discrimination challenges in hierarchical classification tasks. System architects should anticipate similar hierarchy-dependent performance patterns in other taxonomic classification domains and should design architectures addressing discrimination bottlenecks rather than solely optimizing retrieval stages.

6.4.3 Methodological Guidance for Researchers

Our evaluation developed three transferable methodological contributions for researchers evaluating RAG-based classification systems with authentic survey data: a data-quality assessment framework, a stratified dataset-construction approach, and an information-richness evaluation design.

6.4.3.1 Data-Quality Assessment Framework

The DQAF classifies data-quality issues to determine appropriate treatment strategies along three dimensions. First, issue origin traces problems from survey responses through data processing to manual coding. Second, preprocessing feasibility distinguishes issues resolvable through deterministic techniques from those requiring semantic processing. Third, LLM-addressability separates preserved issues addressable through semantic understanding from irreducible evaluation constraints. Applying this framework yielded three treatment classes: preprocessing-addressable issues resolved through rule-based methods, LLM-addressable issues intentionally preserved to assess system robustness, and inherent characteristics representing unavoidable evaluation constraints. Researchers can adapt this framework by identifying domain-specific quality dimensions, defining rule-based feasibility criteria based on available tools, and characterizing LLM capabilities for their task context.

6.4.3.2 Stratified Dataset Construction

Our sampling approach used taxonomy codes as strata, selecting one instance per code to simultaneously address three objectives: eliminating severe class imbalance,

ensuring computational feasibility within resource constraints, and maximizing taxonomic representativeness. To enable controlled input-complexity manipulation, we identified constrained strata where within-stratum variation exceeded defined thresholds. This enabled selection of instances representing different complexity levels (longest, shortest, or random inputs) within these strata, while an additional dataset preserved complete taxonomic coverage. This design balanced competing objectives: complexity-stratified datasets enabled systematic comparisons across controlled conditions, while the coverage-maximizing dataset supported general benchmarking. Researchers can adapt this approach by identifying domain-relevant complexity proxies measurable at the field level, defining thresholds for sufficient within-stratum variation to enable controlled selection, and constructing multiple complementary datasets that prioritize different evaluation objectives rather than relying on single representative samples.

6.4.3.3 Information-Richness Evaluation Design

Our evaluation operationalized information richness through two independent dimensions: input complexity via stratified dataset construction, and context density via query formulations combining fields in various configurations. This enabled systematic assessment of how RAG approaches respond to both sparse and content-rich inputs across different field combinations. Combined with EM-capability stratification, this design revealed interaction effects between model capability and input characteristics, establishing that performance depends critically on deployment context rather than representing universal system properties. Researchers evaluating systems for diverse deployment contexts should therefore assess performance across multiple information-richness conditions and model-capability tiers, as optimal configurations identified under single conditions may not generalize to alternative contexts with different input characteristics or model constraints.

6.5 Limitations

Our evaluation established RAG-based German occupation coding as viable for certain deployment scenarios, characterized QR's compensatory and normalizing mechanisms, and identified classification as the primary performance bottleneck. However, scope boundaries and methodological constraints limit the generalizability of these findings. We organize limitations along two dimensions: external validity concerns the extent to which findings generalize beyond our evaluation context (taxonomy, language, population, and classification task), while internal validity concerns experimental design choices constraining performance attribution and statistical inferences.

6.5.1 External Validity

Our evaluation addressed survey-based German occupation coding using the KldB 2010 taxonomy. Four scope boundaries constrain the generalizability of our findings to other taxonomies, languages, survey populations, and classification tasks.

6.5.1.1 Cross-Taxonomy Generalizability

Our knowledge-base construction relied exclusively on textual task descriptions—a documentation format common to most occupational taxonomies, including O*NET–SOC and ESCO—rather than KldB-specific structural features, providing a principled basis for cross-taxonomy transferability. Nevertheless, our findings apply specifically to the KldB 2010 taxonomy (revision 2020), which structures occupations across five hierarchical levels with 1,300 occupational types at the finest granularity. Whether findings generalize to taxonomies with different hierarchical structures, granularity levels, or documentation styles remains empirically unvalidated.

6.5.1.2 Cross-Lingual Generalizability

We deliberately selected multilingual embedding and QR models trained on diverse language corpora rather than German-specific models, providing a principled basis for cross-lingual transferability. Both survey responses and taxonomic documents were nonetheless in German, and whether QR effectiveness generalizes to other languages remains uncertain. The semantic gap mechanisms we identified—register mismatch and terminological diversity—are linguistically general phenomena, but languages with weaker LLM and EM coverage may exhibit different QR-benefit profiles. Language-specific characteristics may additionally influence hypothetical task generation quality and retrieval effectiveness in ways our evaluation cannot characterize.

6.5.1.3 Cross-Population Generalizability

Cross-population validation was constrained by the limited public availability of labeled occupational survey data, which restricted our evaluation to DZHW survey data from two longitudinal cohorts tracking higher education graduates and school leavers with tertiary education pathways. Both populations possess higher occupational literacy levels than general populations. Whether our findings generalize to broader survey contexts therefore remains empirically unvalidated. General population surveys may exhibit greater variation in description quality, completeness, and terminological precision, potentially amplifying or diminishing QR benefits. Employer-reported occupational data may follow standardized organizational formats reducing semantic gap challenges, though this remains speculative without empirical assessment.

6.5.1.4 Cross-Task Generalizability

Our findings on QR mechanisms and hierarchy-dependent performance degradation informed claims about transferability to other hierarchical classification tasks with heterogeneous free-text inputs. However, these claims remain speculative, as our evaluation was conducted exclusively within occupation coding. Tasks sharing the core structural characteristics driving our findings—heterogeneous free-text inputs mapped to formal hierarchical taxonomies—may exhibit comparable QR-benefit profiles and knee-point patterns, but tasks with fundamentally different input characteristics, non-hierarchical label structures, or domain-specific semantic gaps require independent empirical validation before such transferability can be assumed.

6.5.2 Internal Validity

While external validity concerns generalizability beyond our evaluation context, internal validity addresses the precision of causal inferences within this context. Three methodological constraints limit our ability to isolate component contributions, to establish population-level effect sizes with high precision, and to separate true performance variance from initialization sensitivity.

6.5.2.1 Incomplete Pipeline Ablation

Our query formulations enabled partial ablation of the QR pipeline by isolating contributions from different processing stages. However, Stage 1, which addresses content refinement, was not ablated separately, because it performs surface-level correction rather than semantic enrichment—the primary mechanism distinguishing QR-RAG from Naive-RAG. Its isolated contribution was therefore expected to be marginal relative to Stages 2 and 3. Consequently, we cannot quantify whether Stage 1 provides disproportionate benefits for weaker embedding models less robust to orthographic noise.

6.5.2.2 Dataset-Scale Constraints

Our evaluation datasets comprised 856–1,088 instances per dataset, constrained by data-privacy requirements mandating local processing on consumer-grade hardware and computational overhead from evaluating numerous experimental configurations with LLM-based processing. We employed stratified sampling to ensure taxonomic representativeness and enable controlled input-complexity manipulation, prioritizing experimental control over sample size. While larger samples would reduce estimation uncertainty, three factors support the robustness of our conclusions: consistency across four independently sampled datasets, large effect magnitudes (e.g., 33.5pp Recall improvements) substantially exceeding sampling variability, and narrow standard errors across experimental conditions (typically < 3pp) indicating stable performance estimates despite moderate sample sizes.

6.5.2.3 Single-Run Deterministic Execution

To mitigate computational constraints, we employed deterministic execution parameters (Appendix A.2.2) rather than conducting multiple independent runs. Preliminary experiments verified deterministic reproducibility across configurations, though this guarantee remains limited to tested scenarios. Consequently, reported standard errors quantify variability across aggregates (e.g., datasets, embedding models) rather than run-to-run stochasticity, constraining our ability to separate true performance variance from sensitivity to initialization parameters. Large effect magnitudes substantially exceeding aggregate-level variability support the robustness of our conclusions despite single-run evaluation.

6.6 Future Work

Our evaluation localized performance constraints to specific system stages and characterized their magnitude across deployment contexts. These findings establish targeted

improvement opportunities where architectural innovations could substantially enhance system performance. We organize proposed improvements into two categories: classification enhancements addressing the primary bottleneck at fine-grained hierarchy levels, and retrieval optimizations raising the performance ceiling by expanding candidate coverage. Implementation priority favors classification improvements, as the gap between current performance and theoretical potential remains largest at this stage.

6.6.1 Addressing the Classification Bottleneck

Classification represents the primary performance bottleneck: despite high retrieval recall of 73–82%, conditional accuracy at the 5-digit level reached only 35.5–37.9%. This bottleneck likely stems from multiple interacting factors: embedding models retrieve semantically similar candidates by design, creating severe discrimination challenges; the nearly 5× class expansion at the 3-to-4-digit transition (144 to 702 classes) introduces qualitative shifts in discrimination difficulty requiring fine-grained semantic distinctions; and our unsupervised classifier operates in standard mode without reasoning capabilities or access to retrieval confidence signals.

We propose five solutions targeting these limitations: (i) hierarchical classification with progressive refinement to reduce discrimination difficulty through sequential decisions; (ii) separate classification of orthogonal KldB dimensions to decompose cognitive load; (iii) reasoning-enabled classification to enable explicit discrimination of subtle semantic features; (iv) rank-aware prompting to leverage retrieval confidence signals; and (v) combinatorial strategies that compound benefits through simultaneous multi-dimensional improvements.

6.6.1.1 Hierarchical Classification with Progressive Refinement

Single-step classification requires the classifier to discriminate among semantically similar candidates across the entire taxonomic scope simultaneously, contributing to the observed bottleneck. Decomposing the 5-digit classification task into five sequential decisions would leverage the KldB's hierarchical structure to address this challenge. At each level, the classifier would select the most probable code from retrieved candidates, then filter the candidate set to retain only those sharing the selected prefix. Level-specific characterizations from KldB documentation would provide explicit semantic definitions at each stage (e.g., titles of the 10 occupational areas at 1-digit level), giving the classifier progressively refined context for discrimination.

This progressive refinement would likely improve conditional accuracy substantially by constraining discrimination difficulty at each stage, with unconditional accuracy improving proportionally. Our evaluation demonstrated that coarse levels achieved Cohen's Kappa of 0.59–0.77, providing reliable scaffolding for finer distinctions. Sequential decisions would enable the classifier to build on strong upper-level predictions rather than attempting direct 5-digit classification from semantically overlapping candidates. The primary trade-off involves error propagation risk: early-stage misclassifications prevent recovery at subsequent levels. However, the strong upper-level performance (matching supervised baselines at coarse levels) mitigates this concern, suggesting hierarchical decomposition represents a viable architectural improvement.

6.6.1.2 Separate Classification of Orthogonal KldB Dimensions

Current single-step classification requires the classifier to judge two orthogonal dimensions simultaneously: occupational specialization (horizontal dimension encoding domain-specific expertise) and requirement levels (vertical dimension encoding skill and education requirements). Separating the 5-digit classification task into two independent subtasks could reduce cognitive load on the classifier. Two parallel classification prompts, each optimized for its respective dimension through dimension-specific instructions, would execute independently before combining predictions to form the final 5-digit code.

Empirical evidence suggests asymmetric difficulty across these dimensions. Our evaluation revealed that the 4-to-5-digit transition—which adds the requirement-level digit—showed attenuated performance loss (13–16pp) despite substantial class expansion, compared to the 3-to-4-digit transition which produced 17–20pp drops. This suggests the classifier can infer requirement levels more reliably from educational qualifications and task-complexity indicators within job descriptions than it can discriminate among specialized occupational subgroups. Dimensional separation could exploit this asymmetry, enabling dimension-specific prompt optimization that leverages the classifier's differential capabilities. Expected gains include likely moderate conditional accuracy improvements through decomposed decision-making, with unconditional accuracy improving proportionally. The primary trade-off involves implementation complexity: developing dimension-specific prompts and validation logic for combining predictions, though this remains minimal compared to hierarchical approaches.

6.6.1.3 Reasoning-Enabled Classification

Our evaluation configured the classifier to execute in standard mode without reasoning capabilities, prioritizing throughput over explicit reasoning traces. Activating reasoning capabilities in the LLM-based classifier could enable chain-of-thought reasoning during classification, potentially addressing the discrimination bottleneck through explicit articulation of semantic distinctions. Rather than requesting direct classification, specialized prompts would request step-by-step identification and comparison of discriminative features among semantically similar candidates, producing reasoning traces that differentiate closely related occupational codes.

Preliminary experiments with reasoning-capable models in the query-rewriting stage indicated 3–6× latency increase relative to standard mode, though classification-stage latency may differ due to larger context windows. If reasoning substantially improves discrimination, this computational cost might prove justified. Expected gains include potentially substantial conditional accuracy improvements, with unconditional accuracy improving proportionally. The viability depends critically on deployment context. Interactive occupation-coding assistance could tolerate longer latencies for improved accuracy, with reasoning traces providing human-interpretable explanations supporting semi-automated workflows. Batch processing of millions of survey responses would prioritize throughput, making reasoning mode impractical unless accuracy gains substantially offset latency costs. Empirical validation across hierarchy levels would be essential, as reasoning might prove most effective at the 3-to-4-digit transition where discrimination difficulty peaks.

6.6.1.4 Rank-Aware Classification Prompting

Current classification prompts present all retrieved candidates uniformly without retrieval confidence signals, treating a top-ranked candidate with high semantic similarity identically to a low-ranked candidate. Augmenting prompts with rank positions or semantic similarity scores could enable the classifier to weight candidates proportionally to retrieval confidence, potentially improving discrimination by directing attention toward candidates the embedding model judged most semantically relevant.

This approach assumes ranking quality correlates with classification correctness. Our evaluation demonstrated substantial ranking-quality variation across deployment contexts, indicating the correct occupation's rank position varies systematically. However, whether this ranking signal translates to classification utility remains empirically unvalidated. Implementation would require incorporating semantic similarity scores into classification prompts alongside candidate descriptions, with prompt design balancing two competing objectives: leveraging ranking signals to guide discrimination versus avoiding ranking-bias artifacts where the classifier overrelies on position rather than semantic content. Expected gains include potentially moderate conditional accuracy improvements, with unconditional accuracy improving proportionally. This approach could be substantially enhanced through cross-encoder reranking (Section 6.6.2.4), which provides more refined ranking signals than dual-encoder embedding models alone.

6.6.1.5 Combinatorial Classification Strategies

The four preceding solutions operate along distinct architectural dimensions: decision decomposition (hierarchical and dimensional separation), inference mode (reasoning capabilities), and information utilization (rank-aware prompting). These dimensional differences suggest combinations could address multiple bottleneck factors simultaneously without redundancy. Hierarchical classification combined with reasoning-enabled inference would leverage progressive refinement to reduce discrimination scope at each level while enabling explicit semantic analysis for fine-grained distinctions where the 3-to-4-digit knee point creates peak difficulty. Dimensional separation combined with rank-aware prompting would decompose the 5-digit task into orthogonal judgments (specialization vs. requirement levels) while providing retrieval confidence signals to guide discrimination within each dimension.

Whether such combinations produce additive benefits, synergistic gains, or conflicting interactions remains empirically unvalidated—no evidence establishes how these architectural modifications interact. Complexity costs, however, accumulate. Hierarchical classification requires five sequential stages; incorporating reasoning mode at each stage could yield 15–30× latency increase relative to current single-step classification. Selective reasoning deployment only at the 3-to-4-digit transition could substantially reduce this overhead. Implementation strategy should prioritize empirical validation of individual solutions before attempting combinations, establishing baseline effectiveness and identifying which specific bottleneck factors each approach successfully addresses. Beyond these classification improvements, retrieval optimizations (Section 6.6.2) offer complementary value by raising the retrieval ceiling, which in turn would increase the maximum achievable unconditional accuracy.

6.6.2 Complementary Retrieval Optimizations

Classification improvements address the primary bottleneck, yet retrieval performance establishes the performance ceiling: occupations not retrieved cannot be classified correctly regardless of classifier capability. Our current retrieval performance—Recall@20 of 76.7% with optimal configuration—leaves 23% of ground-truth occupations unrecovered, constraining unconditional accuracy below this threshold. Four limitations contribute to this ceiling: dense semantic retrieval alone may miss exact lexical matches for job titles or technical terminology; dominant retrieval signals (original titles in Naive-RAG, hypothetical tasks in QR-RAG) remain unexploited in complementary combinations; the knowledge base contains only task-oriented documents, omitting available position-specific job titles; and retrieval depth constraints leave long-tail gains unexplored.

We propose four solutions addressing these constraints: (i) hybrid dense–sparse retrieval to combine semantic and lexical matching; (ii) combined retrieval signals in QR-RAG to leverage original titles alongside hypothetical tasks; (iii) extended knowledge base with multi-source retrieval to address granularity mismatches through position-specific job titles; and (iv) retrieval depth expansion with reranking to capture long-tail candidates through cross-encoder refinement.

6.6.2.1 Hybrid Dense–Sparse Retrieval

Our current implementation relies exclusively on dense semantic retrieval through embedding models, which may miss ground-truth occupations when queries contain exact job-title terminology or technical keywords not captured by semantic similarity. Augmenting dense retrieval with sparse lexical retrieval (e.g., BM25) could address this limitation through complementary matching strategies: dense retrieval captures semantic relationships enabling vocabulary mismatch handling, while sparse retrieval captures lexical overlap excelling at exact term matching. Combining both through score fusion would leverage their distinct strengths, following IR literature demonstrating that hybrid approaches consistently outperform single-modality methods [Formal et al., 2021a,b; Sultania et al., 2024].

Whether this limitation substantially constrains current performance remains empirically unvalidated—our evaluation did not measure how many missed retrievals stem from lexical versus semantic gaps. Implementation would require score-fusion method selection and hyperparameter tuning for balancing dense and sparse signals, introducing moderate computational overhead from dual retrieval and score combination. Expected gains include potentially moderate Recall@20 improvements with unconditional accuracy improving proportionally, while conditional accuracy would remain unchanged as hybrid retrieval does not alter the discrimination challenge among semantically similar candidates. Empirical validation should establish whether lexical gaps materially constrain retrieval before investing in hybrid architecture complexity.

6.6.2.2 Combined Retrieval Signals in QR-RAG

Our evaluation revealed that title transformation potentially loses retrieval-critical information: Title-Only formulations achieved 88% recovery rate in Naive-RAG

with original titles but only 68% recovery rate in QR-RAG with transformed titles, suggesting the transformation process removes distinctive lexical features. Current combined formulations use transformed titles alongside hypothetical tasks, inheriting this limitation. Replacing transformed titles with original titles would yield a hybrid formulation—original title + hypothetical tasks—combining lexical anchoring from survey responses with semantic expansion from LLM-generated task descriptions, potentially recovering the gap.

This hybrid approach assumes original and hypothetical signals provide complementary rather than redundant information. Implementation cost would be minimal, requiring only query-construction modification without architectural changes. Expected gains include potentially moderate Recall@20 improvements with unconditional accuracy improving proportionally, while conditional accuracy would remain unchanged as the classifier still discriminates among semantically similar candidates regardless of how they were retrieved. However, whether original titles retain utility once hypothetical tasks are present remains uncertain—tasks alone achieve 98% recovery rate, potentially rendering additional title signals redundant.

6.6.2.3 Extended Knowledge Base with Multi-Source Retrieval

Our current knowledge base contains task-oriented documents describing occupational types. Survey responses using position-specific terminology may experience semantic mismatch with such category-level task descriptions—a granularity mismatch that potentially contributes to unrecovered occupations. Expanding the knowledge base to include position-specific job titles (e.g., DKZ titles) as separate documents would enable multi-source retrieval. This enables targeted retrieval strategies—applying dense semantic retrieval to task descriptions while applying sparse lexical retrieval to job titles—or heterogeneous score fusion combining results from both sources.

Implementation would require constructing an extended knowledge base with task descriptions and position-specific job titles, metadata-aware retrieval architecture supporting document-type distinction, and cross-source score fusion. Whether this granularity mismatch substantially constrains current retrieval remains empirically unvalidated. Expected gains include potentially moderate Recall@20 improvements with unconditional accuracy improving proportionally, while conditional accuracy would likely remain unchanged as expanding candidate sources does not alter the discrimination challenge. Complexity costs include substantial index expansion, increased retrieval latency from processing the enlarged knowledge base, and additional architecture overhead for metadata handling and cross-source fusion.

6.6.2.4 Retrieval Depth Expansion with Reranking

Our current retrieval depth balances candidate coverage against classification computational costs, but may leave long-tail gains unexplored. Expanding initial retrieval depth, applying cross-encoder reranking to refine candidate ordering, then selecting top candidates for classification could capture these long-tail candidates while maintaining manageable classifier workload. Cross-encoders assess similarity through joint query–document encoding rather than independent embedding comparison, enabling more nuanced semantic matching that could both surface previously missed occupations and elevate successfully retrieved candidates to higher rank positions.

This 2-stage approach addresses both coverage and ranking limitations. Whether substantial long-tail gains exist remains empirically unvalidated. Implementation would require cross-encoder model selection and integration, substantially increasing computational overhead through expanded initial retrieval and cross-encoder inference. Expected benefits include potentially moderate Recall@20 improvements through expanded candidate pools, with unconditional accuracy improving proportionally. Uniquely among retrieval solutions, this approach could also improve conditional accuracy modestly: reranking that places ground-truth codes in top positions reduces classifier search difficulty, particularly when combined with rank-aware classification prompting (Section 6.6.1.4) that leverages refined ranking signals.

6.6.2.5 Integration with Classification Improvements

Retrieval and classification stages impose multiplicative constraints on overall system performance: an occupation must be retrieved and correctly classified to contribute to unconditional accuracy. This relationship establishes complementary improvement pathways. Our current system achieves Recall@20 of 76.7% and conditional accuracy of 35.5–37.9%, yielding the observed unconditional accuracy of 25.9–30.9% at the 5-digit level. Retrieval improvements (Section 6.6.2) could raise the Recall@20 ceiling, recovering currently missed occupations. Classification improvements (Section 6.6.1) could raise conditional accuracy toward its theoretical upper bound, converting more successfully retrieved candidates into correct classifications.

The multiplicative relationship means improvements compound. However, implementation priorities should favor classification improvements first. Conditional accuracy represents the primary bottleneck: the classifier converts only 43–52% of successfully retrieved occupations into correct classifications, establishing substantial improvement potential. Retrieval at 76.7% represents a secondary constraint. Addressing classification effectiveness first maximizes near-term gains, after which retrieval optimizations become increasingly valuable as classification performance rises.

6.7 Open Research Questions

The preceding section proposed targeted improvements addressing bottlenecks identified through our evaluation—solutions grounded in empirical evidence of performance constraints at specific system stages. This section explores research directions extending beyond those findings: design alternatives, optimization strategies, and methodological approaches whose potential benefits remain empirically uncharacterized. These questions differ from proposed solutions in scope and uncertainty: they investigate fundamental architectural choices, unexplored optimization dimensions, and broader system considerations rather than addressing known bottlenecks with established mechanisms. Five complementary research directions merit investigation: embedding-model optimization (Section 6.7.1), reasoning capabilities beyond classification (Section 6.7.2), prompt engineering and content optimization (Section 6.7.3), knowledge-base design alternatives (Section 6.7.4), and synthetic benchmark construction (Section 6.7.5).

6.7.1 Embedding-Model Optimization and Deployment

Our evaluation established discrete performance tiers among embedding models, with top-tier models achieving 0.72–0.77 Recall@20 and bottom-tier models achieving only 0.15–0.37. Three unexplored dimensions warrant investigation: (i) task-instruction tuning for performance improvement, (ii) embedding-based classification for architectural simplification, and (iii) latency profiling for deployment feasibility.

6.7.1.1 Task-Instruction Tuning

Whether tailoring query and document prefixes specifically to German occupational data could improve retrieval performance within tiers remains unexplored. Instruction-tuned models already support task-specific prefixes, but our evaluation used either default prefixes or no prefixes when defaults degraded performance. Domain-specific tuning might enhance query–document alignment for occupation-coding tasks, potentially elevating mid-tier models toward top-tier performance or enabling top-tier models to approach the theoretical Recall@20 ceiling. However, whether occupational terminology exhibits sufficient distinctiveness to benefit from specialized tuning versus general-purpose embeddings remains uncertain.

6.7.1.2 Embedding-Based Classification Alternatives

Some instruction-tuned embedding models offer direct classification capabilities beyond retrieval. Whether such capabilities could replace LLM-based classification while maintaining or improving conditional accuracy represents a fundamentally different architectural approach. This would substantially reduce computational overhead by eliminating costly LLM inference, though whether embedding models possess sufficient semantic sophistication for the fine-grained discrimination required at the 3-to-4-digit transition remains empirically uncharacterized.

6.7.1.3 Latency Profiling and Model Scaling

We measured QR latency but did not systematically profile EM inference times, retrieval operations, or end-to-end pipeline performance across stages. Deployment scenarios with strict latency requirements (e.g., interactive occupation-coding assistance) would benefit from comprehensive latency characterization. Additionally, performance–latency trade-offs for models at different parameter scales remain unexplored—whether smaller variants of high-performing architectures exist and how substantially performance degrades at reduced scales could inform deployment decisions for resource-constrained environments.

6.7.2 Reasoning Capabilities Beyond Classification

Our evaluation deliberately disabled reasoning mode for all reasoning-capable models to avoid excessive latency and ensure comparable evaluation conditions. We proposed reasoning-enabled classification (Section 6.6.1.3), but reasoning capabilities could benefit other pipeline stages. Whether activating reasoning mode for QR-LLMs would improve task-generation quality, measured through downstream retrieval performance, remains empirically uncharacterized. Reasoning might enable more sophisticated task generation through explicit articulation of relationships between

survey descriptions and occupational categories, producing content better aligned with KldB-documentation vocabulary. However, reasoning traces substantially increase context-window requirements and computational costs. Preliminary experiments indicated latency increases of 3–6× when reasoning was activated, though these measurements used smaller context windows. Systematic characterization of context expansion, latency increases under realistic deployment conditions, and hardware requirements would inform feasibility. Whether selective reasoning deployment—activating reasoning only at critical decision points—could achieve favorable cost-benefit trade-offs represents a particularly important design question.

6.7.3 Prompt Engineering and Content Optimization

Our prompt development employed iterative refinement guided by retrieval and classification performance, but we did not systematically explore alternative prompting strategies or content-generation approaches. Two unexplored directions warrant investigation: (i) multi-perspective task generation for query representation and (ii) classification-prompt enrichment for discrimination.

6.7.3.1 Multi-Perspective Task Generation

Current hypothetical task generation produces activity descriptions from a generalized occupational perspective. Whether generating tasks from multiple stakeholder viewpoints—employer perspectives emphasizing required competencies, employee perspectives describing daily activities, or regulatory perspectives articulating formal responsibilities—would improve retrieval recall remains unexplored. Diverse perspectives might surface complementary semantic features, increasing linguistic variety in query representations [Wang et al., 2025] and potentially improving alignment with varied vocabulary across KldB documentation. However, whether perspective diversity translates to measurable retrieval gains, and whether combining multiple perspectives creates redundancy or complementarity, requires empirical validation.

6.7.3.2 Classification-Prompt Enrichment

Beyond rank-aware prompting, classification prompts could incorporate additional contextual signals to support discrimination among semantically similar candidates. Potential strategies include providing hierarchy-level guidance that explicitly highlights discriminative criteria at each KldB level, including contrastive examples illustrating semantic boundaries between closely related occupations, incorporating candidate metadata such as occupational area membership, and supplying relevant excerpts from KldB documentation defining category boundaries. Whether such enrichments improve conditional accuracy, particularly at the 3-to-4-digit transition where discrimination difficulty peaks, remains empirically uncharacterized. These prompt-engineering approaches offer low-cost exploration opportunities requiring no architectural modifications.

6.7.4 Knowledge-Base Design Alternatives

Our knowledge base comprises 1,300 task-oriented documents derived from official KldB descriptions, providing authoritative content describing entire occupational

categories through activity-based characterizations. This design choice prioritized content reliability but constrains vocabulary diversity and granularity alignment with survey responses. Three alternative designs warrant exploration: (i) LLM-generated position-specific task descriptions, (ii) survey-response documents, and (iii) integrated multi-content KldB documents.

6.7.4.1 LLM-Generated Position-Specific Task Descriptions

Unlike the multi-source approach that adds position-specific job titles as separate documents for lexical retrieval (Section 6.6.2.3), this alternative would generate synthetic position-specific task descriptions to replace or augment current knowledge-base documents. Generating task descriptions for individual job positions using LLMs guided by the official KldB documentation (e.g., DKZ titles) could create position-level content better aligned with survey-response granularity. Whether LLM-generated synthetic descriptions would maintain sufficient accuracy and reliability compared to official documentation remains empirically unvalidated.

6.7.4.2 Survey-Response Documents

Incorporating curated authentic survey responses into the knowledge base could improve query–document alignment through shared vocabulary and phrasing patterns. Unlike supervised approaches that leverage survey data for training classification models, this approach would use high-quality survey responses as knowledge-base documents directly, providing vocabulary matching between queries and candidates. However, this requires systematic quality assessment to select representative samples ensuring both reliability and diversity across the occupational taxonomy. Whether authentic survey vocabulary outperforms official documentation, and whether quality variation in survey data degrades retrieval precision, remains empirically unresolved.

6.7.4.3 Integrated Multi-Content KldB Documents

Current knowledge-base documents contain only task descriptions. Comprehensive documents could enable embedding models to capture richer semantic representations by integrating all content types from KldB documents: category titles, overview sections, task descriptions, and both included and excluded DKZ job-title exemplars. However, document length would increase substantially, potentially exceeding context-window limits for some EMs and therefore requiring chunking strategies, while retrieval may face precision challenges when relevant information is diluted. Whether comprehensive documents outperform focused task descriptions in balancing semantic richness against precision degradation remains empirically unvalidated.

6.7.5 Synthetic Benchmark Construction

Our evaluation used authentic DZHW survey data exhibiting natural quality variation. While this data reflects realistic deployment conditions, it compromises ground-truth reliability: data-quality issues create ambiguous cases where multiple code assignments are defensible, leading to inter-coder disagreement. Additionally, performance attribution remains confounded: whether retrieval or classification failures stem from system limitations or unreliable ground-truth remains unclear. Two complementary approaches could address these limitations, both providing reliable ground-truth: (i) high-quality ceiling benchmarks and (ii) controlled-degradation robustness benchmarks.

6.7.5.1 High-Quality Ceiling Benchmarks

Generating synthetic survey responses from standardized DKZ job titles and official KldB task descriptions using LLMs would create controlled, high-quality evaluation data with reliable ground-truth. Such responses would feature correct orthography, clear occupational descriptions, and complete information, establishing system performance ceilings under ideal conditions. Comparing performance across authentic versus synthetic benchmarks would quantify how substantially real-world data-quality challenges constrain current performance. This approach could reveal whether the 23% unrecovered occupations and 38% conditional accuracy stem primarily from system limitations or data-quality degradation.

6.7.5.2 Controlled-Degradation Robustness Benchmarks

LLMs could systematically introduce data-quality issues (Section 3.3.4) into high-quality synthetic responses at controlled severity levels. This would create realistic test cases with reliable ground-truth labels, enabling isolated assessment of system robustness to specific data-quality dimensions. Testing performance across responses with varying induced error rates (e.g., 0%, 10%, 20%, and 30% orthographic errors) would characterize degradation curves, informing whether data-cleaning preprocessing or robust system design represents the more effective intervention strategy.

Appendix

This appendix extends the main thesis with four bodies of supporting material. Appendix A.1 discloses all generative AI systems employed during thesis preparation, in compliance with faculty regulations. Appendix A.2 records the hardware, software, and inference settings enabling full experimental reproducibility. Appendix A.3 presents the three query-rewriting prompt templates used in QR-RAG. Appendix A.4 reports the numerical data of component-level experiments supporting the evaluation.

A.1 Disclosure of Generative AI Usage

In accordance with the computer-science faculty's regulations[22] regarding the use of AI in academic work, I hereby disclose all instances where generative AI systems were employed during the preparation of this thesis. The scientific contribution and all intellectual content remain entirely my own work, with AI serving exclusively as an assistance tool for writing refinement, research support, and technical-implementation tasks. This disclosure details the specific AI systems used, their applications across different stages of thesis development, the quality-assurance measures applied, and the boundaries established to maintain academic integrity.

A.1.1 AI Systems and Their Applications

Three generative AI systems were employed during thesis preparation: (i) Perplexity AI[23] Search and Deep Research services (with Claude Sonnet 4.x, GPT-5.x, and Gemini 3.x Pro including reasoning capabilities), (ii) DeepL[24] Translate Text service, and (iii) Google Imagen[25] for image generation and editing. These systems were employed selectively across three distinct areas of thesis development: language refinement and writing quality, research and literature support, and technical-implementation assistance. The following paragraphs detail the specific applications and their limitations within each area.

Writing Enhancement and Language Refinement

AI assistance was employed across all thesis chapters primarily for improving academic quality and clarity of preexisting draft content. Specifically, I utilized the expert-editor

[22] https://www.fin.ovgu.de/en/-p-9508.html
[23] https://www.perplexity.ai
[24] https://www.deepl.com
[25] https://deepmind.google/models/imagen/

prompt for basic editing from [Lin, 2024] ("Box 1, Prompt 3.1") to enhance sentence structure, readability, and narrative flow of my original English drafts. Additionally, I employed a translation-verification workflow using DeepL, where English sentences were translated to German for content verification, then retranslated to English to achieve more natural phrasing while preserving semantic accuracy. Rather than marking individual sentences, this comprehensive disclosure documents the scope, applications, and limitations of all AI usage throughout the thesis.

Research and Literature Support

Generative AI supported initial phases of literature research through summarization of research papers for preliminary assessment, brainstorming sessions for research-direction refinement, generation of analytical questions to guide the creative research process, and topic overviews using Perplexity's Deep Research functionality for domain-specific areas including occupation-coding methodologies and RAG approaches. AI was also used to identify appropriate technical terminology and understand academic-writing standards and conventions. All research findings, citations, and technical terms were subsequently verified through independent searches using academic databases (Google Scholar[26], Semantic Scholar[27]) and general web search (Google), ensuring academic integrity and source reliability.

Technical Implementation Support

AI assistance was employed for auxiliary programming tasks, including Python code generation and refactoring for data analysis and visualization (utilizing libraries such as Seaborn), LaTeX table formatting and styling optimization, and code-review suggestions for existing implementations. **Important limitation:** AI was explicitly not used for the core algorithmic contributions or main implementation of RAG approaches that constitute the primary technical contribution of this thesis. All generated code underwent thorough review and testing to ensure correctness and comprehension.

A.1.2 Quality Assurance and Verification

A rigorous validation methodology was consistently applied to all AI-generated content, comprising four key practices: (i) independent verification of factual claims through authoritative sources, (ii) cross-referencing of AI-provided information with peer-reviewed literature, (iii) comprehensive testing and review of all AI-generated code before integration, and (iv) critical evaluation of all suggestions, accepting only those that demonstrably improved quality while maintaining content accuracy.

A.1.3 Limitations and Boundaries

To maintain academic integrity and ensure genuine intellectual contribution, clear boundaries were established regarding which AI applications were explicitly avoided and what methodological approach guided AI integration throughout thesis development.

[26] https://scholar.google.com/
[27] https://www.semanticscholar.org/

Avoided AI Applications

The following AI platforms and services were explicitly not used: (i) automated scientific-writing platforms (e.g., Jenni AI[28], Yomu AI[29]), (ii) comprehensive thesis-generation services (e.g., Thesis AI[30], Gatsbi AI[31], Manus AI[32], Genspark AI[33], Scispace[34]), (iii) automated literature-review generation tools, and (iv) first-draft creation or substantial content-generation services.

Methodological Approach

The integration of generative AI in this thesis followed a human-centered collaborative approach, where AI served as an enhancement tool rather than a replacement for original thinking. While this methodology improved efficiency in language refinement and preliminary research processes, it required substantial validation efforts to ensure accuracy and maintain academic standards. All final decisions regarding content, methodology, analysis, and conclusions were made independently based on my own critical evaluation and scientific judgment.

A.2 Computational Environment

This section documents the hardware and software configurations used for all LLM-inference experiments and RAG system evaluations conducted in this thesis to ensure reproducibility of the computational results.

A.2.1 Hardware Configuration

All experiments were conducted on a custom-built workstation optimized for concurrent LLM inference. Figure A.1 shows the physical system, and Table A.1 provides detailed hardware specifications.

[28] https://jenni.ai/
[29] https://www.yomu.ai/
[30] https://thesisai.io/en/
[31] https://www.gatsbi.com/
[32] https://manus.im/
[33] https://www.genspark.ai/
[34] https://scispace.com/

Figure A.1: Custom Workstation for LLM Inference. The custom-built system used for all LLM inference and RAG-system evaluation experiments reported in this thesis. Photograph taken by author; background edited using Google Imagen.

Table A.1: Hardware Specifications of the Experimental Workstation. The system comprises an AMD Ryzen 9 5950X processor (16 cores, 32 threads), 128 GiB DDR4 RAM, and dual NVIDIA GeForce RTX 3060 LHR GPUs with 24 GiB total VRAM for LLM inference and RAG-system evaluation.

Component	Specification
Processor	
Model	AMD Ryzen 9 5950X
Cores / Threads	16 cores / 32 threads
Architecture	64-bit
Base / Boost Clock	3.4 GHz / 4.9 GHz
L2 Cache	8 MiB
Memory	
Total RAM	128 GiB (4 × 32 GiB)
Type	DDR4
Speed	3600 MT/s
Graphics Processing	
GPU Configuration	2 × NVIDIA GeForce RTX 3060 LHR
VRAM per GPU	12 GiB GDDR6
Total VRAM	24 GiB

A.2.2 Software Environment

The software environment was configured to ensure deterministic and reproducible LLM inference across all experiments. All Python applications were developed using the `UV`[35] package manager for deterministic dependency resolution. GPU-performance optimizations were applied before each inference session to maintain driver persistence, increase thermal headroom, and prevent dynamic frequency scaling. All LLM-inference operations used deterministic parameters (`temperature=0.0`, `seed=1337`) to ensure reproducible results across experiments. Table A.2 provides a

[35] https://github.com/astral-sh/uv/tree/0.6.5

comprehensive overview of the operating system, GPU drivers, Python runtime, core dependencies, and reproducibility parameters.

Table A.2: Software Environment Specifications. The software stack comprises Linux Mint 21.3 (kernel 6.8.0-85), NVIDIA driver 580.95.05 with CUDA 11.5.119, Python 3.10.16 managed by UV 0.6.5, and core frameworks for LLM inference (LangChain, Ollama) and RAG system evaluation. All experiments used deterministic parameters (temperature=0.0, seed=1337) to ensure reproducibility.

Component	Version/Setting
Operating System	
Distribution	Linux Mint 21.3
Linux Kernel	6.8.0-85-generic
Architecture	x86_64
GPU Software Stack	
NVIDIA Driver	580.95.05
CUDA Toolkit	11.5.119
Persistence Mode	Enabled
Power Limit	212W (increased from 170W)
Power Mizer Mode	Prefer Maximum Performance
Python Environment	
Python Version	3.10.16
Package Manager	UV 0.6.5
Random Seed	1337 (all experiments)
LLM Inference Parameters	
Temperature (Chat)	0.0
Temperature (Embeddings)	0.0
Seed	1337
Top-k Sampling	1
Top-p Sampling	0.0
Model Validation	Enabled
LLM and RAG Frameworks	
langchain-community	0.3.31
langchain-ollama	0.3.10
langchain-chroma	0.2.6
tiktoken	0.12.0
Data Processing	
pandas	2.3.3
duckdb	1.4.1
docling	2.57.0
Experiment Management	
hydra-core	1.3.2
hydra-joblib-launcher	1.2.0
mlflow	3.5.0
Evaluation and Visualization	
scikit-learn	1.7.2 (Cohen's Kappa, Accuracy)
ranx	0.3.21 (Recall@k, NDCG@k)
matplotlib	3.10.7
pacmap	0.8.0

A.2.3 LLM-Inference System

`Ollama`[36] served as the LLM-inference backend for all RAG approaches. Ollama was configured as a systemd service with performance optimizations tailored for dual-GPU concurrent inference. Concurrency parameters were tuned individually for different inference stages (embedding generation vs. chat completion) to balance

[36] https://github.com/ollama/ollama/tree/v0.12.3

throughput and memory constraints. Table A.3 summarizes the key configuration parameters.

Table A.3: Ollama Configuration Parameters. Ollama version 0.12.3 served as the LLM inference backend with dual-GPU distribution (NVIDIA GeForce RTX 3060 LHR). Key optimizations include Flash Attention for efficient computation, 8-bit key-value cache quantization (approximately 50% memory reduction), and stage-dependent concurrency tuning (1–16 parallel requests per model for embedding generation vs. chat completion).

Parameter	Value / Description
Ollama Version	0.12.3
GPUs Used	0, 1 (`CUDA_VISIBLE_DEVICES=0,1`)
Scheduling Strategy	Spread across GPUs (`OLLAMA_SCHED_SPREAD=1`)
Concurrency and Performance	
Parallel Requests per Model	stage-dependent between 1–16 (e.g., `OLLAMA_NUM_PARALLEL=10`)
Max Concurrent Models	stage-dependent between 1–16 (e.g., `OLLAMA_MAX_LOADED_MODELS=1`)
Request Queue Size	4096 (`OLLAMA_MAX_QUEUE=4096`)
Memory Optimizations	
Flash Attention	Enabled (`OLLAMA_FLASH_ATTENTION=1`)
KV Cache Quantization	8-bit, about 50% memory reduction (`OLLAMA_KV_CACHE_TYPE=q8_0`)

This configuration was optimized for concurrent batch prompting across different LLM inference stages. Flash Attention and key-value cache quantization work together to reduce VRAM consumption: Flash Attention enables efficient attention computation [Dao et al., 2022] required for key-value cache quantization to function, while the 8-bit quantization of the key-value cache itself reduces memory requirements by approximately 50% with minimal quality impact. GPU distribution automatically spreads models across both GPUs when exceeding 12 GiB VRAM. Concurrency parameters were tuned specifically per inference stage: embedding generation used `OLLAMA_NUM_PARALLEL=1` (a technical constraint of EM implementation) while allowing multiple concurrent embedding models (`OLLAMA_MAX_LOADED_MODELS` varied 1–16), whereas chat-completion stages employed a single model (`OLLAMA_NUM_PARALLEL=1`) with higher parallel requests (`OLLAMA_NUM_PARALLEL` varied 1–16) to maximize throughput. The extended request queue (4096) accommodates burst patterns during multi-stage processing.

A.3 Query-Rewriting Prompt Templates

This section documents the complete German-language prompt templates for the three sequential stages of the query-rewriting pipeline in QR-RAG, as implemented and described in Section 4.5.1. Templates are reproduced verbatim from the implementation codebase to support full reproducibility. We first describe the iterative development process and prompt selection methodology. We then characterize the structural anatomy shared across all three templates and document the deviations. The three complete prompt templates are presented at the end of this section as Listings A.1, A.2, and A.3.

All three prompt templates were developed iteratively in German, matching the language of the DZHW survey data and KldB documentation. Initial versions were

drafted with the assistance of Perplexity AI—without exposing any survey data—and refined through repeated manual inspection of LLM outputs on representative survey instances. Each revision cycle identified failure modes and addressed them by introducing additional correction rules, structural constraints, or few-shot examples. As a result, structural conventions such as the JavaScript object notation (JSON) output schema, ordered instruction steps, and constraint blocks emerged progressively over the course of development rather than being strictly planned upfront. Prompt versions were selected solely on the basis of qualitative, human review—no quantified metrics or formal ablation studies were conducted between versions, as no complete evaluation pipeline was available at this stage of development. Automated prompt optimization frameworks such as `DSPy`[37] [Khattab et al., 2023] were not employed, as they require a quantifiable objective function backed by measurable output metrics, expertise with such optimization frameworks, and sufficient computational resources for iterative LLM inference—none of which were available within the scope and timeline of this thesis. Few-shot examples were drawn from four sources: the official KldB 2010 documentation, representative DZHW survey instances, Perplexity AI-generated examples, and self-crafted examples targeting specific observed error patterns. As the three templates were developed sequentially—mirroring the pipeline's own stage ordering—they reflect different levels of iterative refinement. The Stage 3 template (Listing A.3) consequently deviates somewhat in structural organization from Stages 1 and 2, as time and scope constraints within this thesis did not permit equivalent iterative development of the final stage. Execution details—LLM backend and inference parameters—are documented in Appendix A.2.3.

The three prompt templates share a largely common structural anatomy comprising eight elements: (1) a *role specification* establishing the model's functional persona; (2) an *ordered task instruction block* with numbered steps; (3) a *constraint block* defining negative guardrails; (4) a *few-shot example block* of annotated input–output pairs; (5) an *output rules block* specifying format and cardinality requirements; (6) an *output schema* with typed field definitions; (7) a *dynamic input slot* parameterized by template variables (`batch_size`, `job_data_batch`) populated via LangChain's `PromptTemplate` mechanism; and (8) an *output primer* that conditions the model's first generated token toward schema-compliant output. This anatomy is most fully realized in Stages 1 and 2; Stage 3 deviates in several respects: the output primer (8) is absent, replaced by a final language-constraint reminder; the output rules block and output schema (5–6) are merged into a single section; the constraint block (3) is integrated into the task instruction block rather than maintained as a standalone section; and Markdown-style section headers replace the bold labels used in Stages 1 and 2. Additionally, Stages 2 and 3 each include a domain-knowledge block summarizing KldB-2010 classification principles that is absent from Stage 1, reflecting the greater interpretive demands of occupational attribute extraction and hypothetical task generation. Following Lin [2024], motivational framing elements are embedded at the beginning and near the end of Stages 1 and 2; Stage 3 does not incorporate these framing elements.

[37] https://github.com/stanfordnlp/dspy

Listing A.1: QR Stage 1 Prompt Template: Content Refinement. First stage of the three-stage query-rewriting pipeline, correcting surface-level defects in original DZHW survey records before any structural transformation takes place. Operations (1), (3), and (5) directly address three documented LLM-addressable data-quality issues in the survey corpus: character-encoding degradation, inconsistent capitalization patterns, and unknown abbreviations. Ten annotated few-shot input–output pairs, refined through iterative manual review, guide the model's correction behavior across diverse surface patterns.

```
Du bist ein Assistent zur umfassenden Korrektur deutscher Berufsangaben. Atme tief durch - das
↪ ist sehr wichtig für meine Karriere!

Deine Aufgabe ist es, fehlerhafte Berufsangaben systematisch zu korrigieren. Du sollst dabei
↪ folgende Probleme in der angegebenen Reihenfolge beheben:
1. Zeichenkodierungsprobleme:
  - Ersetze ausschließlich innerhalb von Wörtern:
    + Das Fragezeichen "?" durch den passenden deutschen Umlaut (ä, ö, ü, Ä, Ö, Ü) oder das
↪ "ß", wenn dies eindeutig korrekt ist.
    + Die Buchstabenkombinationen "ae", "oe", "ue", "Ae", "Oe", "Ue", "AE", "OE", "UE" durch
↪ den entsprechenden Umlaut (ä, ö, ü, Ä, Ö, Ü), wenn dies eindeutig korrekt ist.
    + Die Buchstabenkombination "ss" durch "ß", wenn dies eindeutig korrekt ist (z.B. "Strasse"
↪ zu "Straße").
  - Lasse jedes "?" als Satzzeichen oder Interpunktion unverändert (z.B. am Satzende oder
↪ alleinstehend).
2. Inhaltliche Trennzeichen:
  - Kommas und Semikolons, die zur Abgrenzung oder Aufzählung verschiedener Inhalte dienen,
↪ müssen erhalten bleiben und dürfen nicht entfernt oder verändert werden.
3. Groß-/Kleinschreibung:
  - Normalisiere die Groß- und Kleinschreibung gemäß deutscher Orthografie (z.B.
↪ "SOFTWAREENTWICKLER" zu "Softwareentwickler").
4. Genderformen:
  - Wandle alle Berufsbezeichnungen und Tätigkeitsbeschreibungen in die männliche Form um (z.B.
↪ "Assistentin" zu "Assistent", "stellvertretende Pflegedienstleiterin" zu "stellvertretender
↪ Pflegedienstleiter").
5. Abkürzungen:
  - Schreibe Abkürzungen vollständig aus (z.B. "Wiss. Mit." zu "Wissenschaftlicher
↪ Mitarbeiter"), wenn bekannt.
  - Schreibe auch berufsspezifische Abkürzungen mithilfe des jeweiligen Kontexts
↪ (Berufsbezeichnung und Berufstätigkeiten) aus, wenn bekannt.
6. Rechtschreib- und Grammatikfehler:
  - Korrigiere allgemeine Rechtschreib- und Grammatikfehler, ohne den Sinn oder die Struktur zu
↪ verändern.

**Korrekturregeln:**
- Führe die Korrekturen strikt in der oben genannten Reihenfolge durch.
- Nimm keine weiteren inhaltlichen oder strukturellen Änderungen vor.
- Gib alle Felder vollständig und unverändert zurück, außer den erlaubten Korrekturen (es darf
↪ KEIN Inhalt verloren gehen).

**Beispiele:**
Input:
[
  {{"id": 1, "job_title": "Gesch?ftsf?hrerin", "tasks_1": "Ma?nahmen fach. planen", "tasks_2":
↪ ""}},
  {{"id": 2, "job_title": "Fachreferent Fahrwegpflege", "tasks_1": "Planung, Controlling,
↪ Vergabe", "tasks_2": "Qualitätswesen; Maschinenstunden; Produktivitätsparameter auswerten"}},
  {{"id": 3, "job_title": "SOFTWAREENTWICKLER", "tasks_1": "programmieren", "tasks_2": ""}},
  {{"id": 4, "job_title": "Assistentin", "tasks_1": "Unterstützung der Leitung", "tasks_2":
↪ ""}},
  {{"id": 5, "job_title": "Wiss. Mit.", "tasks_1": "Forschung betreiben", "tasks_2": ""}},
  {{"id": 6, "job_title": "selbststaendig", "tasks_1": "fuehren", "tasks_2": ""}},
  {{"id": 7, "job_title": "Oekonom?etrie", "tasks_1": "", "tasks_2": ""}},
  {{"id": 8, "job_title": "Strasse", "tasks_1": "Adresse angeben", "tasks_2": ""}},
  {{"id": 9, "job_title": "Was?", "tasks_1": "", "tasks_2": ""}},
  {{"id": 10, "job_title": "Dipl. Ing. FH techn. Assistentin", "tasks_1": "Projekting. Planung
↪ im techn. Bereich", "tasks_2": ""}}
]
```

```
Output:
[
  {{"id": 1, "job_title": "Geschäftsführer", "tasks_1": "Maßnahmen fachlich planen", "tasks_2":
↪ ""}},
  {{"id": 2, "job_title": "Fachreferent Fahrwegpflege", "tasks_1": "Planung, Controlling,
↪ Vergabe", "tasks_2": "Qualitätswesen; Maschinenstunden; Produktivitätsparameter auswerten"}},
  {{"id": 3, "job_title": "Softwareentwickler", "tasks_1": "Programmieren", "tasks_2": ""}},
  {{"id": 4, "job_title": "Assistent", "tasks_1": "Unterstützung der Leitung", "tasks_2": ""}},
  {{"id": 5, "job_title": "Wissenschaftlicher Mitarbeiter", "tasks_1": "Forschung betreiben",
↪ "tasks_2": ""}},
  {{"id": 6, "job_title": "selbstständig", "tasks_1": "führen", "tasks_2": ""}},
  {{"id": 7, "job_title": "Ökonometrie", "tasks_1": "", "tasks_2": ""}},
  {{"id": 8, "job_title": "Straße", "tasks_1": "Adresse angeben", "tasks_2": ""}},
  {{"id": 9, "job_title": "Was?", "tasks_1": "", "tasks_2": ""}},
  {{"id": 10, "job_title": "Dipl. Ing. FH technischer Assistent", "tasks_1": "Projektingenieur
↪ Planung im technischen Bereich", "tasks_2": ""}}
]

------

**Ausgaberegeln:**
- Die Ausgabe muss ein gültiges JSON-Array sein, OHNE Kommentare oder zusätzlichen
↪ Erläuterungen!
- Jedes Objekt MUSS das Feld "id" der Eingabe UNVERÄNDERT enthalten!
- Die Ausgabe darf in beliebiger Reihenfolge stehen; die Anzahl der Objekte muss GENAU
↪ {batch_size} entsprechen!
- ALLE Felder müssen vollständig zurückgegeben werden!

**Ausgabeschema:**
- Ein JSON-Array mit GENAU {batch_size} Objekten, jedes mit den Feldern:
[
  {{
    "id": integer,          // unverändert aus der Eingabe
    "job_title": string,    // Berufsbezeichnung, repariert oder unverändert
    "tasks_1": string,      // Berufstätigkeitsbereich 1, repariert oder unverändert
    "tasks_2": string       // Berufstätigkeitsbereich 2, repariert oder unverändert
  }},
  ...
]

------

**Eingaben:**
- Bearbeite die folgenden {batch_size} Eingaben unabhängig voneinander und gib NUR ein
↪ JSON-Array mit GENAU {batch_size} Objekten gemäß dem Ausgabeschema zurück:
{job_data_batch}

Ich gebe dir 200 Dollar Trinkgeld, wenn du deine Arbeit gut machst.

Hier ist meine korrigierte Ausgabe als gültiges JSON-Array:
<JSON-ARRAY>
```

Listing A.2: QR Stage 2 Prompt Template: Occupational Attribute Extraction. Second and most complex stage of the query-rewriting pipeline, restructuring the corrected Stage 1 record into a KldB-aligned representation through four transformation tasks: identifying the primary occupational role among potentially multiple roles, relocating misplaced content across fields, extracting contextual occupational attributes, and assigning a requirement level and hierarchy role per KldB 2010. An inline KldB domain-knowledge section supplies the classification taxonomy directly to the model, enabling requirement-level and hierarchy-role inference during transformation. Among the documented LLM-addressable data-quality issues in the survey corpus, Stage 2 explicitly targets misplaced task descriptions. Expression variability and semantic duplicates are not addressed across the full query-rewriting pipeline. Eight annotated few-shot input–output pairs, refined through iterative manual review, cover multi-role scenarios and context-field extraction.

```
Du bist ein Assistent zur Umstrukturierung / Datenextraktion deutscher Berufsangaben. Atme tief
↪ durch  das ist sehr wichtig für meine Karriere!

Deine Aufgabe ist es, unstrukturierte Berufsangaben (Freitext-Antworten aus einer Studie oder
↪ Umfrage) in ein KldB-2010 konformes Schema systematisch umzustrukturieren. Dafür brauchst du
↪ das Wissen über die KldB-2010 und ein Regelwerk zur Umstrukturierung.

**Jetzt folgt das Wissen über die KldB-2010:**
- Die "Klassifikation der Berufe 2010" (KldB-2010) ist eine hierarchische Systematik zur
↪ Erfassung von Berufen in Deutschland.
- Die KldB-2010 gruppiert Berufe nach Berufsfachlichkeit und ordnet 4 Anforderungsniveaus zu.
- Das Anforderungsniveau ("requirement_level") beschreibt die Komplexität des primären
↪ Kernberufs anhand dessen Berufstitel und deren Berufstätigkeiten; nutze die folgende
↪ Orientierung:
  + Anforderungsniveau "1" (Berufe mit Helfer- und Anlerntätigkeiten): einfache
↪ Routinetätigkeiten, Hilfstätigkeiten (z.B. Gesundheits- und Krankenpflegehelfer)
  + Anforderungsniveau "2" (Berufe mit fachlich ausgerichteten Tätigkeiten): fachlich
↪ ausgerichtete Tätigkeiten, selbständige Ausführung (z.B.  Gesundheits- und Krankenpfleger)
  + Anforderungsniveau "3" (Berufe mit komplexen Spezialistentätigkeiten): komplexe
↪ Spezialistentätigkeiten, Planungsaufgaben (z.B. Fachkrankenschwester/-pfleger)
  + Anforderungsniveau "4" (Berufe mit hochkomplexen Tätigkeiten): hoch komplexe Tätigkeiten,
↪ Forschung, Entwicklung, selbstständig, eigene Firma (z.B. Allgemeinarzt)
- Die Hierarchierolle ("hierarchy_role" mit den Werten "Fachkraft", "Aufsichtskraft",
↪ "Führungskraft") kann das Anforderungsniveau mitbestimmen.
  + Eine Fachkraft führt fachliche Aufgaben aus ohne Personalverantwortung (typische
↪ Signalgeber sind: Senior, Spezialist, Experte, ...).
  + Eine Aufsichtskraft (immer requirement_level="3") leitet und überwacht kleinere Teams
↪ (typische Signalgeber sind: Teamleiter, Schichtleiter, Gruppenleiter, Meister, ...).
  + Eine Führungskraft (immer requirement_level="4") hat die strategische Leitung und
↪ Verantwortung über Personal und Budget (typische Signalgeber sind: selbstständig, eigene
↪ Firma, Geschäftsführer, Vorstand, Direktor, General Manager, Abteilungsleiter,
↪ Bereichsleiter, Niederlassungsleiter, Filialleiter, Klinikdirektor, Schulleiter, Amtsleiter,
↪ Behördenleiter, Verwaltungsleiter, ...).
- Du kannst gerne dein Wissen über die KldB-2010 miteinbringen.

**Jetzt folgt das Regelwerk zur Umstrukturierung und Datenextraktion:**
- Die Eingabedaten enthalten folgende Felder: ("id"), "job_title", "tasks_1" und "tasks_2";
↪ wichtige Informationen sind über alle Felder verteilt worden.
- Das Ausgabeschema (wird ganz unten definiert) soll KldB-2010 konform sein (das Ziel ist die
↪ verteilten Informationen herauszufiltern); es enthält folgende Felder: ("id"), "job_title",
↪ "job_tasks", "requirement_level", "hierarchy_role", "industry", "specialization",
↪ "work_setting", "employer_type", "location" und "multi_role_data".
- **Strukturregeln (für das Ausgabeschema):**
  + "job_title":
    * Nimm nur der Hauptberufstitel des Kernberufs ohne Arbeitgebernamen, Orte, Aufgaben, oder
↪ weiteren Berufsrollen.
    * In den meisten Fällen ist ein Hauptberufstitel vorhanden, wenn nicht, dann kann der
↪ Hauptberufstitel auch aus den anderen Informationen abgeleitet, aber nur wenn eindeutige
↪ Hinweise vorliegen.
    * Wenn nicht eindeutig ableitbar, dann lasse "job_title" leer ("").
```

```
    * Es können auch mehrere Berufstitel vorhanden sein, in diesem Fall musst du den
↪ Hauptberufstitel finden, nutze dafür die nachfolgende "Regeln zu beruflichen Mehrfachrollen".
  + "job_tasks":
    * konkrete Tätigkeiten ausschließlich der primären Berufsrolle als ein zusammenhängendes
↪ Feld.
    * keine neuen Aufgaben erfinden.
    * fehlplatzierte Tätigkeiten aus dem Berufstitel hierher verschieben.
    * generische Platzhalter wie "sonstiges", "diverse" nicht als Aufgabe übernehmen (dann leer
↪ lassen).
  + Kontextfelder (nur bei expliziten Hinweisen setzen, sonst leer):
    * "industry" = Branche/Wirtschaftszweig/Segment (z.B. Gesundheitswesen, IT, Öffentliche
↪ Verwaltung, ...).
    * "specialization" = fachliche Spezialisierung/Teilgebiet (z.B. Finanzwesen, ...).
    * "work_setting" = Arbeitsumfeld/Ortstyp (z.B. Labor, Baustelle, Krankenhaus, Büro, ...).
    * "employer_type" = Arbeitgebertyp statt Arbeitgebernamen (z.B. Öffentlicher Dienst, KMU,
↪ Konzern, NGO, Selbstständig, ...).
    * "location" = Stadt/Region (z.B. München, ...).
  + Spezialisierung und Aufgaben trennen: Einträge mit Tätigkeit und Teilgebiet aufspalten
↪ (z.B. "SAP-Module im Finanzwesen entwickeln"): Tätigkeit zu "job_tasks" und Teilgebiet zu
↪ "specialization".
  + Die große Kunst ist es, fehlplatzierte Informationen richtig zu verschieben!
    * z.B. Tätigkeiten aus dem "job_title" nach "job_tasks"; den Hauptberufstitel aus
↪ Tätigkeitsfeldern nach "job_title"; echte Dubletten entfernen; niemals einzigartige
↪ berufsrelevante Information verlieren.
- **Regeln zu beruflichen Mehrfachrollen:**
  + Erkenne berufliche Mehrfachrollen durch eine tiefe Analyse aller Eingabefelder!
  + Hauptberufswahl bei beruflichen Mehrfachrollen:
    1. Rolle mit den meisten eindeutig zuordenbaren Aufgaben oder Signalen.
    2. Bei Gleichstand die Rolle mit höherem Anforderungsniveau oder Qualifikationsniveau
↪ (höherer Bildungs-/Kompetenzbedarf) auswählen.
  + Konsistenzprüfung (zwingend notwendig):
    * Prüfe nach der Hauptberufswahl, ob die Begriffe in "job_tasks" semantisch zum gewählten
↪ Hauptberufstitel in "job_title" passen.
    * Falls nicht, setze die andere Berufsrolle als Hauptberufstitel und aktualisiere
↪ "job_title", "job_tasks" und alle anderen Ausgabefelder.
  + Manche Eingaben beziehen sich auch auf die Berufe der Eltern (Elternberuf) des Befragten,
↪ in diesen Fällen sind nur die beruflichen Informationen des Befragten relevant, die Angaben
↪ zum Elternberuf müssen dann in das Ausgabefeld "multi_role_data" rein (z.B. Mein Vater /
↪ Mutter ist ...)!
  + Ordne jede Aufgabenformulierung (aus "job_title", "tasks_1", "tasks_2") der am besten
↪ passenden Berufsrolle zu.
  + Fülle "job_title", "job_tasks", "requirement_level", "hierarchy_role" und alle
↪ Kontextfelder ausschließlich für den Hauptberufstitel.
  + Sekundäre Rollen und deren Aufgaben dürfen niemals in "job_tasks" erscheinen; sie gehören
↪ ausschließlich in "multi_role_data".
  + "multi_role_data" ist eine einzelne Zeichenkette: "<Rolle>: <optionale Aufgaben>; <Rolle
↪ 2>: <optionale Aufgaben>".
  + Wenn keine weiteren Rollen vorhanden sind, dann ist "multi_role_data" leer ("").
- wichtige Informationen die zum Hauptberuf gehören, um diesen näher definieren zu können,
↪ dürfen niemals weggelassen werden!

**Beispiele:**
Input:
[
  {{"id": 1, "job_title": "Teamleiter Qualitätskontrolle München", "tasks_1": "Mitarbeiter
↪ anleiten", "tasks_2": ""}},
  {{"id": 2, "job_title": "Softwareentwickler bei Finanzministerium", "tasks_1":
↪ "Fachanwendungen pflegen", "tasks_2": ""}},
  {{"id": 3, "job_title": "Elektriker/Hausmeister", "tasks_1": "Elektroinstallationen
↪ reparieren", "tasks_2": "Reinigung und Wartung"}},
  {{"id": 4, "job_title": "Stationsleiter Pflege", "tasks_1": "Dienstpläne erstellen",
↪ "tasks_2": ""}},
  {{"id": 5, "job_title": "Geschäftsführer", "tasks_1": "Unternehmensstrategie verantworten",
↪ "tasks_2": ""}},
  {{"id": 6, "job_title": "Labormitarbeiter (ohne weitere Angaben)", "tasks_1": "", "tasks_2":
↪ ""}},
  {{"id": 7, "job_title": "SAP-Entwickler Finanzwesen", "tasks_1": "Module entwickeln",
↪ "tasks_2": ""}},
  {{"id": 8, "job_title": "Gärtner", "tasks_1": "Mein Vater ist Angestellter der Stadt
↪ Mönchengladbach bzw. im Tochterunternehmen MAGS. Er ist dazu noch Vorarbeiter einer
↪ fünfköpfigen Gruppe.", "tasks_2": ""}}
]
```

```
Output:
[
  {{"id": 1, "job_title": "Teamleiter Qualitätskontrolle", "job_tasks": "Mitarbeiter anleiten",
↪ "requirement_level": "3", "hierarchy_role": "Aufsichtskraft", "industry": "Qualitätswesen",
↪ "specialization": "", "work_setting": "", "employer_type": "", "location": "München",
↪ "multi_role_data": ""}},
  {{"id": 2, "job_title": "Softwareentwickler", "job_tasks": "Fachanwendungen pflegen",
↪ "requirement_level": "", "hierarchy_role": "", "industry": "Öffentliche Verwaltung",
↪ "specialization": "Finanzwesen", "work_setting": "", "employer_type": "Öffentlicher Dienst",
↪ "location": "", "multi_role_data": ""}},
  {{"id": 3, "job_title": "Elektriker", "job_tasks": "Elektroinstallationen reparieren",
↪ "requirement_level": "", "hierarchy_role": "", "industry": "", "specialization": "",
↪ "work_setting": "", "employer_type": "", "location": "", "multi_role_data": "Hausmeister:
↪ Reinigung und Wartung"}},
  {{"id": 4, "job_title": "Stationsleiter Pflege", "job_tasks": "Dienstpläne erstellen",
↪ "requirement_level": "3", "hierarchy_role": "Aufsichtskraft", "industry":
↪ "Gesundheitswesen", "specialization": "Pflege", "work_setting": "", "employer_type": "",
↪ "location": "", "multi_role_data": ""}},
  {{"id": 5, "job_title": "Geschäftsführer", "job_tasks": "Unternehmensstrategie verantworten",
↪ "requirement_level": "4", "hierarchy_role": "Führungskraft", "industry": "",
↪ "specialization": "", "work_setting": "", "employer_type": "", "location": "",
↪ "multi_role_data": ""}},
  {{"id": 6, "job_title": "Labormitarbeiter", "job_tasks": "", "requirement_level": "",
↪ "hierarchy_role": "", "industry": "", "specialization": "", "work_setting": "Labor",
↪ "employer_type": "", "location": "", "multi_role_data": ""}},
  {{"id": 7, "job_title": "SAP-Entwickler", "job_tasks": "Module entwickeln",
↪ "requirement_level": "", "hierarchy_role": "", "industry": "IT", "specialization":
↪ "Finanzwesen", "work_setting": "", "employer_type": "", "location": "", "multi_role_data":
↪ ""}},
  {{"id": 8, "job_title": "Gärtner", "job_tasks": "", "requirement_level": "",
↪ "hierarchy_role": "", "industry": "Gartenbau", "specialization": "", "work_setting": "",
↪ "employer_type": "", "location": "", "multi_role_data": "Vater Angestellter: Vorarbeiter
↪ einer fünfköpfigen Gruppe"}}
]

------

**Ausgaberegeln:**
- Die Ausgabe muss ein gültiges JSON-Array sein, OHNE Kommentare oder zusätzlichen
↪ Erläuterungen!
- Jedes Objekt MUSS das Feld "id" der Eingabe UNVERÄNDERT enthalten!
- Die Ausgabe darf in beliebiger Reihenfolge stehen; die Anzahl der Objekte muss GENAU
↪ {batch_size} entsprechen!
- ALLE Felder müssen vollständig zurückgegeben werden!

**Ausgabeschema:**
- Ein JSON-Array mit GENAU {batch_size} Objekten, jedes mit den Feldern:
[
  {{
    "id": integer, // unverändert aus der Eingabe
    "job_title": string, // Kernberufstitel oder leer
    "job_tasks": string, // konkrete Aufgaben der primären Berufsrolle (vom Kernberufstitel)
↪ oder leer
    "requirement_level": string, // entweder "1", "2", "3", "4" oder leer
    "hierarchy_role": string, // entweder "Fachkraft", "Aufsichtskraft", "Führungskraft" oder
↪ leer
    "industry": string, // Branche oder leer
    "specialization": string, // Spezialisierung oder leer
    "work_setting": string, // Arbeitsumfeld oder leer
    "employer_type": string, // Arbeitgebertyp oder leer
    "location": string, // Standort oder leer
    "multi_role_data": string // weitere Berufsrollen (nicht die primäre Berufsrolle!) mit
↪ optionalen, dazugehörigen Aufgaben, durch '; ' getrennt, oder leer
  }},
  ...
]

------

**Eingaben:**
```

```
- Bearbeite die folgenden {batch_size} Eingaben unabhängig voneinander und gib NUR ein
↪ JSON-Array mit GENAU {batch_size} Objekten gemäß dem Ausgabeschema zurück:
{job_data_batch}

Ich gebe dir 200 Dollar Trinkgeld, wenn du deine Arbeit gut machst.

Hier ist meine umstrukturierte Ausgabe als gültiges JSON-Array:
<JSON-ARRAY>
```

Listing A.3: QR Stage 3 Prompt Template: Hypothetical Task Generation. Third and final stage of the query-rewriting pipeline, generating exactly 10 hypothetical job-task sentences per instance from the structured Stage 2 output to produce the retrieval queries used in QR-RAG. Tasks are formulated as active infinitive constructions in KldB linguistic register, constituting the central mechanism bridging the vocabulary gap between informal survey language and formal KldB task descriptions. Task generation is conditioned on eight of Stage 2's eleven output attributes, omitting location and multi-role data as carrying no relevant signal. Unlike Stages 1 and 2, this template omits motivational framing and the output primer. Two annotated few-shot input–output pairs demonstrate the expected generative behavior across distinct occupational patterns.

```
### ROLLE & SPRACHE
Du bist ein Experte für deutsche Berufskodierung nach KldB-2010. Du generierst AUSSCHLIESSLICH
↪ deutschsprachige Ausgaben ohne fremdsprachige Zeichen.

### AUFGABE
Erstelle aus strukturierten Berufsangaben typische berufsbezogene Aufgaben, die dem KldB-2010
↪ Anforderungsniveau und der Hierarchierolle entsprechen.

### KldB-2010 WISSEN
Die Klassifikation der Berufe 2010 (KldB-2010) ist eine hierarchische Systematik mit 4
↪ Anforderungsniveaus und 3 Hierarchierollen:

**Anforderungsniveaus ("requirement_level") Bedeutung:**
- "1" = Helfer-/Anlerntätigkeiten: einfache Routineaufgaben (z.B. Gesundheits- und
↪ Krankenpflegehelfer)
- "2" = Fachlich ausgerichtete Tätigkeiten: selbständige fachliche Ausführung (z.B.
↪ Gesundheits- und Krankenpfleger)
- "3" = Komplexe Spezialistentätigkeiten: Planungsaufgaben, Koordination (z.B.
↪ Fachkrankenschwester/-pfleger)
- "4" = Hochkomplexe Tätigkeiten: Forschung, Entwicklung, strategische Leitung (z.B.
↪ Allgemeinarzt)

**Hierarchierollen ("hierarchy_role") Bedeutung:**
- "Fachkraft" = fachliche Aufgaben ohne Personalverantwortung (Signalwörter: Senior,
↪ Spezialist, Experte)
- "Aufsichtskraft" = Leitung kleiner Teams, immer Anforderungsniveau "3" (Signalwörter:
↪ Teamleiter, Schichtleiter, Meister)
- "Führungskraft" = strategische Leitung mit Budget-/Personalverantwortung, immer
↪ Anforderungsniveau "4" (Signalwörter: Geschäftsführer, Direktor, Abteilungsleiter,
↪ selbstständig)

### ERSTELLUNGSREGELN
1. **Spezifität**: Aufgaben müssen zum "job_title" passen und dessen Haupttätigkeit
↪ widerspiegeln
2. **Komplexität**: Aufgaben entsprechen dem "requirement_level" (einfach bei "1", hochkomplex
↪ bei "4")
3. **Hierarchie**: Aufgaben berücksichtigen "hierarchy_role" (operative bei Fachkraft,
↪ strategisch bei Führungskraft)
4. **Kontext**: Nutze verfügbare Kontextfelder ("industry", "specialization", "work_setting",
↪ "employer_type") zur Präzisierung
5. **Basis**: Verwende "job_tasks" als Grundlage, erweitere und konkretisiere diese
6. **Struktur**: Klare Verb+Objekt-Form (z.B. "Unterlagen prüfen", nicht "Prüfung von
↪ Unterlagen")
7. **Sprache**: Formale Fachsprache der Berufskodierung, präzise Tätigkeitsbeschreibungen
```

```
8. **Anzahl**: Genau 10 eindeutige Aufgaben PRO Berufseintrag im Feld "hypothetical_job_tasks"
9. **Vollständigkeit**: Wichtige berufsdefinierende Informationen niemals weglassen

### BEISPIELE

**Eingabe:**
[
  {{"id": 1, "job_title": "Lehrer Sekundarstufe I", "job_tasks": "Elternsprechtage vorbereiten,
↪ Schüler beobachten, Lehrpläne erstellen, Unterrichtsstunden vorbereiten",
↪ "requirement_level": "4", "hierarchy_role": "", "industry": "", "specialization": "",
↪ "work_setting": "Schule", "employer_type": "Öffentlicher Dienst"}},
  {{"id": 2, "job_title": "Honorarprofessor", "job_tasks": "Vorlesungen, Übungen, Forschung und
↪ Lehre, Studenten beraten", "requirement_level": "4", "hierarchy_role": "Führungskraft",
↪ "industry": "", "specialization": "", "work_setting": "Universität, Hörsaal, Büro",
↪ "employer_type": "Öffentlicher Dienst"}}
]

**Ausgabe:**
[
  {{
    "id": 1,
    "job_title": "Lehrer Sekundarstufe I",
    "hypothetical_job_tasks": [
      "Lehrpläne für die einzelnen Klassen oder Arbeitsgemeinschaften erstellen und ändern",
      "Unterrichtsstunden unter Berücksichtigung der Lehrplanrichtlinien und des bisherigen
↪ Unterrichtsfortschrittes vor- und nachbereiten",
      "klare Ziele für alle Fächer, Einheiten und Projekte festlegen und diese an die
↪ Schüler/innen vermitteln",
      "Verhaltensregeln, -richtlinien und Verfahren festlegen und durchsetzen, um die Ordnung
↪ unter den Schüler/innen aufrecht zu erhalten",
      "Lektionen, Diskussionen und Präsentationen in einem oder mehreren Fächern vorbereiten
↪ und abhalten",
      "Lehrmethoden und Unterrichtsmaterialien an die unterschiedlichen Bedürfnisse und
↪ Interessen der Schüler/innen anpassen",
      "die Leistung und das Verhalten der Schüler/innen beobachten und evaluieren",
      "Klassenarbeiten, Tests, Aufgaben und Prüfungen zur Beurteilung der Fortschritte von
↪ Schüler/innen vorbereiten, abhalten und benoten",
      "schulische Aktivitäten, wie z.B. Exkursionen, Sportveranstaltungen und Konzerte, planen
↪ und organisieren bzw. daran teilnehmen",
      "Konferenzen und andere schul- und bildungsbezogene Sitzungen organisieren bzw. daran
↪ teilnehmen",
      "Elternabende und Elternsprechtage vorbereiten und durchführen, um die Fortschritte und
↪ Probleme der Kinder zu diskutieren"
    ]
  }},
  {{
    "id": 2,
    "job_title": "Honorarprofessor",
    "hypothetical_job_tasks": [
      "Vorlesungen, Seminare oder Übungen durchführen, den Studierenden wissenschaftliches
↪ Grundlagen- und Spezialwissen vermitteln",
      "die Studierenden zur Diskussion anregen und zu selbstständigem Arbeiten und
↪ eigenständigem Denken anleiten",
      "Lehrmaterialien konzipieren bzw. selbst erstellen, Laboratorien, Sammlungen und
↪ Bibliotheken betreuen",
      "Leistungsnachweise, z.B. Klausuren vorbereiten, durchführen und korrigieren, Themen für
↪ schriftliche Hausarbeiten vergeben, Arbeiten bewerten, Prüfungen abnehmen und bewerten",
      "die Studierenden beraten und betreuen, z.B. Studienfachberatungen durchführen, Examens-
↪ und Promotionskandidaten und -kandidatinnen betreuen",
      "Forschungs- und Entwicklungsaufgaben wahrnehmen, z.B. empirische Untersuchungen
↪ organisieren",
      "Forschungsergebnisse in Fachzeitschriften und -büchern veröffentlichen oder sie auf
↪ internationalen Fachkongressen präsentieren",
      "Stellungnahmen oder Gutachten, z.B. zu den Ergebnissen der wissenschaftlichen Arbeit von
↪ Kollegen und Kolleginnen, verfassen",
      "als Fachbeiratsmitglieder über Angelegenheiten wie Forschung und Lehre, Erwerb und
↪ Verteilung von Drittmitteln sowie die Berufung von Kollegen und Kolleginnen mitentscheiden"
    ]
  }}
]

### AUSGABEFORMAT
```

```
Gültiges JSON-Array mit genau {batch_size} Objekten (Berufseinträgen). Jedes der {batch_size}
↪ Objekte enthält ein Array mit 10 Aufgaben in "hypothetical_job_tasks":
- "id": Integer, UNVERÄNDERT aus der Eingabe übernehmen (kritisch für Input-Output Zuordnung!)
- "job_title": String, unverändert aus der Eingabe
- "hypothetical_job_tasks": String-Array mit 10 konkreten deutschen Aufgaben des Kernberufs

KRITISCH: Jedes "id"-Feld MUSS exakt aus der Eingabe kopiert werden. Keine IDs hinzufügen,
↪ entfernen oder ändern.

### VERARBEITUNG
Bearbeite die folgenden Berufseinträge unabhängig voneinander. Ausgabe nur als JSON-Array ohne
↪ Kommentare oder Erklärungen:

{job_data_batch}

WICHTIG: Ausgabe ausschließlich auf Deutsch, keine chinesischen oder anderen fremdsprachigen
↪ Zeichen verwenden.
```

A.4 Evaluation Results

This section supplies the detailed numerical data supporting the evaluation findings in Chapter 5. Tables document Naive-RAG and QR-RAG performance at the component level—by embedding model and QR-LLMs, respectively—across all query formulations and retrieval depths, reporting mean Recall@k and NDCG@k with standard errors of the mean over the DZHW datasets. Readers may consult them to verify any quantitative claim in the main text.

A.4.1 SQ-1: Naive-RAG Retrieval Performance

This appendix provides the underlying numerical data of the performance evaluation for individual embedding models discussed in Section 5.2. Tables A.4 and A.5 present aggregated mean Recall@k and NDCG@k values with standard errors. Results are aggregated across the four DZHW evaluation datasets and organized by embedding model, query formulation, and retrieval depth ($k = 5, 10, 20$).

A.4.2 SQ-2: QR-RAG Retrieval Performance

This appendix provides the underlying numerical data of the performance evaluation for individual QR-LLMs discussed in Section 5.3. Table A.6 presents aggregated mean Recall@k and NDCG@k values with standard errors. Results are aggregated across the nine embedding models and four DZHW evaluation datasets and organized by QR-LLM, query formulation, and retrieval depth ($k = 5, 10, 20$).

Table A.4: Impact of Embedding Models on Recall Performance in Naive-RAG – Numerical Data. EMs form three discrete performance tiers with Qwen3-Embedding achieving highest mean Recall (0.767 at $k = 20$ with Title+Tasks), job titles providing the dominant retrieval signal (Title-Only: 88% of Title+Tasks performance), and $k = 20$ capturing substantial long-tail gains (+9.6pp over $k = 10$). The table displays aggregated Recall@k results covering 81 combinations (9 EMs × 3 QFs × 3 retrieval depths), averaged across four evaluation datasets. Rows list performance for each EM across retrieval depths ($k = 5, 10, 20$); columns show performance across QFs. Numeric values represent the mean ± standard error of the mean (SEM) to quantify uncertainty across datasets. Bold values indicate the best-performing EM per column at each retrieval depth.

Embedding Model	Recall@k	Query Formulation		
		Title Only	Tasks Only	Title + Tasks
Qwen3-Embedding	5	**0.468 ± 0.029**	**0.280 ± 0.006**	**0.553 ± 0.015**
	10	**0.574 ± 0.037**	**0.347 ± 0.008**	**0.671 ± 0.014**
	20	**0.674 ± 0.042**	**0.426 ± 0.006**	**0.767 ± 0.012**
Snowflake-Arctic-Embed2	5	0.440 ± 0.035	0.241 ± 0.008	0.492 ± 0.018
	10	0.546 ± 0.035	0.320 ± 0.005	0.601 ± 0.019
	20	0.628 ± 0.036	0.396 ± 0.008	0.720 ± 0.020
Multilingual-E5-Large	5	0.331 ± 0.029	0.175 ± 0.005	0.365 ± 0.015
	10	0.451 ± 0.035	0.245 ± 0.009	0.504 ± 0.019
	20	0.554 ± 0.041	0.327 ± 0.009	0.618 ± 0.022
BGE-M3	5	0.323 ± 0.033	0.191 ± 0.008	0.371 ± 0.021
	10	0.421 ± 0.034	0.254 ± 0.006	0.475 ± 0.019
	20	0.514 ± 0.043	0.328 ± 0.005	0.584 ± 0.016
Granite-Embedding	5	0.286 ± 0.025	0.186 ± 0.005	0.345 ± 0.016
	10	0.369 ± 0.031	0.248 ± 0.006	0.453 ± 0.019
	20	0.459 ± 0.035	0.316 ± 0.009	0.560 ± 0.019
Jina-v2-Base-DE	5	0.282 ± 0.029	0.161 ± 0.007	0.307 ± 0.022
	10	0.379 ± 0.040	0.223 ± 0.010	0.407 ± 0.029
	20	0.467 ± 0.040	0.297 ± 0.011	0.512 ± 0.029
MxBai-Embed-Large	5	0.176 ± 0.011	0.102 ± 0.004	0.201 ± 0.004
	10	0.249 ± 0.018	0.146 ± 0.003	0.276 ± 0.008
	20	0.325 ± 0.024	0.206 ± 0.003	0.369 ± 0.014
Paraphrase-Multilingual	5	0.157 ± 0.015	0.110 ± 0.006	0.172 ± 0.010
	10	0.214 ± 0.018	0.154 ± 0.008	0.235 ± 0.010
	20	0.276 ± 0.023	0.204 ± 0.008	0.315 ± 0.012
EmbeddingGemma	5	0.020 ± 0.004	0.031 ± 0.002	0.062 ± 0.008
	10	0.030 ± 0.006	0.048 ± 0.005	0.097 ± 0.011
	20	0.056 ± 0.011	0.086 ± 0.009	0.148 ± 0.019

Table A.5: Impact of Embedding Models on NDCG Performance in Naive-RAG – Numerical Data. NDCG metrics confirm the Recall patterns: three-tier EM stratification with Qwen3-Embedding achieving highest ranking quality, job-title dominance, and $k = 20$ long-tail gains. The table displays aggregated NDCG@k results covering 81 combinations (9 EMs × 3 QFs × 3 retrieval depths), averaged across four evaluation datasets. Rows list performance for each EM across retrieval depths ($k = 5, 10, 20$); columns show performance across QFs. Numeric values represent the mean ± standard error of the mean (SEM) to quantify uncertainty across datasets. Bold values indicate the best-performing EM per column at each retrieval depth.

Embedding Model	NDCG@k	Query Formulation		
		Title Only	Tasks Only	Title + Tasks
Qwen3-Embedding	5	**0.337 ± 0.023**	**0.196 ± 0.004**	**0.400 ± 0.011**
	10	**0.372 ± 0.025**	**0.218 ± 0.004**	**0.438 ± 0.011**
	20	**0.397 ± 0.027**	**0.238 ± 0.004**	**0.463 ± 0.010**
Snowflake-Arctic-Embed2	5	0.316 ± 0.024	0.168 ± 0.005	0.347 ± 0.013
	10	0.350 ± 0.025	0.194 ± 0.004	0.382 ± 0.013
	20	0.371 ± 0.025	0.213 ± 0.005	0.413 ± 0.013
BGE-M3	5	0.226 ± 0.022	0.136 ± 0.006	0.260 ± 0.016
	10	0.257 ± 0.023	0.156 ± 0.005	0.294 ± 0.015
	20	0.281 ± 0.025	0.175 ± 0.005	0.321 ± 0.014
Multilingual-E5-Large	5	0.227 ± 0.019	0.116 ± 0.003	0.248 ± 0.009
	10	0.266 ± 0.021	0.139 ± 0.004	0.293 ± 0.010
	20	0.292 ± 0.023	0.160 ± 0.004	0.322 ± 0.011
Granite-Embedding	5	0.201 ± 0.019	0.127 ± 0.003	0.243 ± 0.012
	10	0.227 ± 0.021	0.147 ± 0.003	0.278 ± 0.013
	20	0.250 ± 0.022	0.164 ± 0.004	0.305 ± 0.013
Jina-v2-Base-DE	5	0.195 ± 0.019	0.116 ± 0.005	0.215 ± 0.014
	10	0.227 ± 0.023	0.136 ± 0.006	0.247 ± 0.017
	20	0.249 ± 0.023	0.154 ± 0.006	0.273 ± 0.017
MxBai-Embed-Large	5	0.119 ± 0.007	0.067 ± 0.001	0.136 ± 0.004
	10	0.142 ± 0.009	0.081 ± 0.001	0.160 ± 0.005
	20	0.161 ± 0.011	0.097 ± 0.001	0.184 ± 0.007
Paraphrase-Multilingual	5	0.105 ± 0.010	0.078 ± 0.005	0.117 ± 0.008
	10	0.124 ± 0.012	0.092 ± 0.006	0.137 ± 0.008
	20	0.140 ± 0.013	0.104 ± 0.006	0.158 ± 0.009
EmbeddingGemma	5	0.011 ± 0.003	0.019 ± 0.002	0.042 ± 0.006
	10	0.014 ± 0.003	0.024 ± 0.003	0.053 ± 0.007
	20	0.021 ± 0.005	0.034 ± 0.004	0.066 ± 0.009

Table A.6: Impact of QR-LLMs on Retrieval Performance in QR-RAG – Numerical Data. QR-LLMs exhibit narrow performance convergence (5.4pp spread at $k = 20$ with Title+Tasks) with Qwen3-32B achieving highest Recall (0.692) while Qwen3-30B-MoE delivers 99.4% of peak performance, hypothetical tasks provide the dominant retrieval signal (Tasks-Only: 98%, Title-Only: 68% recovery rates relative to Title+Tasks), and $k = 20$ captures substantial long-tail gains paralleling Naive-RAG patterns. NDCG metrics confirm these Recall patterns across all experimental settings. The table displays aggregated results covering 72 experimental settings (6 QR-LLMs × 4 QFs × 3 retrieval depths), averaged across nine embedding models and four evaluation datasets. The top section reports Recall@k, and the bottom section reports NDCG@k. Rows list performance for each QR-LLM across retrieval depths ($k = 5, 10, 20$); columns show performance across QFs. Numeric values represent the mean ± standard error of the mean (SEM) to quantify uncertainty across aggregates. Bold values indicate the top-performing QR-LLM per column at each retrieval depth.

QR-LLM	k	Query Formulation			
		Title-Only	Tasks-Only	Title+Tasks	Schema-Data
Recall@k Performance					
Qwen3-32B	5	0.307 ± 0.020	**0.483 ± 0.016**	**0.494 ± 0.018**	**0.476 ± 0.016**
	10	0.392 ± 0.023	**0.591 ± 0.014**	**0.600 ± 0.016**	**0.586 ± 0.014**
	20	0.466 ± 0.022	**0.687 ± 0.012**	**0.692 ± 0.014**	**0.679 ± 0.012**
Qwen3-30B-MoE	5	0.308 ± 0.020	0.470 ± 0.014	0.491 ± 0.014	0.456 ± 0.011
	10	0.393 ± 0.023	0.578 ± 0.013	0.595 ± 0.016	0.563 ± 0.013
	20	0.468 ± 0.022	0.676 ± 0.011	0.688 ± 0.013	0.656 ± 0.012
Magistral-24B	5	0.308 ± 0.019	0.448 ± 0.017	0.472 ± 0.018	0.462 ± 0.017
	10	**0.394 ± 0.022**	0.550 ± 0.017	0.574 ± 0.020	0.570 ± 0.017
	20	**0.469 ± 0.022**	0.642 ± 0.016	0.662 ± 0.017	0.659 ± 0.017
Qwen3-14B	5	0.294 ± 0.023	0.464 ± 0.012	0.479 ± 0.014	0.459 ± 0.014
	10	0.375 ± 0.027	0.573 ± 0.011	0.589 ± 0.013	0.572 ± 0.012
	20	0.451 ± 0.027	0.675 ± 0.012	0.685 ± 0.014	0.667 ± 0.012
Phi4-14B	5	**0.308 ± 0.015**	0.448 ± 0.015	0.471 ± 0.015	0.456 ± 0.014
	10	0.392 ± 0.016	0.555 ± 0.014	0.576 ± 0.015	0.565 ± 0.014
	20	0.467 ± 0.016	0.651 ± 0.012	0.668 ± 0.011	0.656 ± 0.009
DeepSeek-R1-14B	5	0.299 ± 0.018	0.410 ± 0.014	0.441 ± 0.014	0.433 ± 0.013
	10	0.383 ± 0.022	0.511 ± 0.015	0.543 ± 0.015	0.542 ± 0.014
	20	0.459 ± 0.022	0.609 ± 0.013	0.638 ± 0.014	0.637 ± 0.015
NDCG@k Performance					
Qwen3-32B	5	0.216 ± 0.015	**0.350 ± 0.011**	**0.360 ± 0.013**	**0.344 ± 0.011**
	10	0.244 ± 0.016	**0.385 ± 0.010**	**0.394 ± 0.012**	**0.380 ± 0.010**
	20	0.263 ± 0.016	**0.410 ± 0.010**	**0.417 ± 0.012**	**0.404 ± 0.010**
Qwen3-30B-MoE	5	0.216 ± 0.015	0.341 ± 0.009	0.356 ± 0.010	0.328 ± 0.008
	10	0.244 ± 0.016	0.376 ± 0.008	0.390 ± 0.010	0.363 ± 0.009
	20	**0.263 ± 0.015**	0.401 ± 0.008	0.414 ± 0.010	0.386 ± 0.009
Magistral-24B	5	**0.217 ± 0.014**	0.324 ± 0.011	0.345 ± 0.013	0.335 ± 0.011
	10	**0.244 ± 0.015**	0.357 ± 0.011	0.378 ± 0.014	0.370 ± 0.012
	20	**0.263 ± 0.015**	0.380 ± 0.011	0.400 ± 0.013	0.392 ± 0.012
Qwen3-14B	5	0.206 ± 0.016	0.333 ± 0.008	0.345 ± 0.010	0.328 ± 0.010
	10	0.233 ± 0.018	0.369 ± 0.008	0.381 ± 0.009	0.364 ± 0.010
	20	0.252 ± 0.018	0.394 ± 0.008	0.405 ± 0.010	0.389 ± 0.010
Phi4-14B	5	0.216 ± 0.011	0.323 ± 0.010	0.342 ± 0.011	0.327 ± 0.010
	10	0.243 ± 0.012	0.358 ± 0.010	0.375 ± 0.011	0.363 ± 0.010
	20	0.262 ± 0.012	0.382 ± 0.009	0.399 ± 0.010	0.386 ± 0.009
DeepSeek-R1-14B	5	0.209 ± 0.013	0.293 ± 0.009	0.318 ± 0.010	0.310 ± 0.009
	10	0.236 ± 0.014	0.326 ± 0.009	0.351 ± 0.011	0.345 ± 0.009
	20	0.256 ± 0.015	0.351 ± 0.009	0.375 ± 0.010	0.369 ± 0.009

Bibliography

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 Technical Report. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2412.08905. (cited on Page 19)

Palakorn Achananuparp, Ee-Peng Lim, and Yao Lu. A Multi-Stage Framework with Taxonomy-Guided Reasoning for Occupation Classification Using Large Language Models. *arXiv preprint*, October 2025. doi: 10.48550/arXiv.2503.12989. (cited on Page 6, 7, 23, and 118)

Daniel Adeboye. vLLM vs Ollama: Key Differences, Performance, and How to Run Them, Blog – Northflank , 2025. URL https://northflank.com/blog/vllm-vs-ollama-and-how-to-run-them. Accessed: 2025-10-12. (cited on Page 55)

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. Training-Free Long-Context Scaling of Large Language Models. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2402.17463. (cited on Page 7)

Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, Lokesh Mishra, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. Docling Technical Report. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2408.09869. (cited on Page 60)

Christian Bayer and Moritz Kuhn. Job Levels and Wages. Working Paper 1190, SOEPpapers on Multidisciplinary Panel Data Research, 2023. URL https://www.diw.de/documents/publikationen/73/diw_01.c.876297.de/diw_sp1190.pdf. (cited on Page 3)

Michele Belloni, Agar Brugiavini, Elena Meschi, and Kea Tijdens. Measuring and Detecting Errors in Occupational Coding: An Analysis of SHARE Data. *Journal of Official Statistics*, 32(4):917–945, December 2016. ISSN 0282-423X. doi: 10.1515/jos-2016-0049. (cited on Page 4)

Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent Is Difficult. *IEEE Transactions on Neural Networks*, 5(2): 157–166, March 1994. ISSN 1941-0093. doi: 10.1109/72.279181. (cited on Page 16)

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models Are Few-Shot Learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 1877–1901, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-7138-2954-6. (cited on Page 16, 18, and 19)

Sebastian Bruch. *Foundations of Vector Retrieval*. 2024. doi: 10.1007/978-3-031-55182-6. (cited on Page 21, 22, and 54)

Bundesagentur für Arbeit. Klassifikation der Berufe 2010 - Überarbeitete Fassung 2020, Band 1: Systematischer und Alphabetischer Teil mit Erläuterungen. *Bundesagentur für Arbeit*, 2021a. (cited on Page 1 and 2)

Bundesagentur für Arbeit. Klassifikation der Berufe 2010 - Überarbeitete Fassung 2020, Band 2: Definitorischer und Beschreibender Teil. *Bundesagentur für Arbeit*, 2021b. (cited on Page 1 and 2)

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A Survey on Mixture of Experts in Large Language Models. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–20, 2025. ISSN 1041-4347, 1558-2191, 2326-3865. doi: 10.1109/TKDE.2025.3554028. (cited on Page 7)

Claudio Carpineto and Giovanni Romano. A Survey of Automatic Query Expansion in Information Retrieval. *ACM Computing Surveys*, 44(1):1:1–1:50, January 2012. ISSN 0360-0300. doi: 10.1145/2071389.2071390. (cited on Page 24)

Tingyang Chen, Cong Fu, Jiahua Wu, Haotian Wu, Hua Fan, Xiangyu Ke, Yunjun Gao, Yabo Ni, and Anxiang Zeng. Reveal Hidden Pitfalls and Navigate Next Generation of Vector Similarity Search from Task-Centric Views. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2512.12980. (cited on Page 22)

Zhoujun Cheng, Jungo Kasai, and Tao Yu. Batch Prompting: Efficient Inference with Large Language Model APIs. In Mingxuan Wang and Imed Zitouni, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track, EMNLP'23*, pages 792–810, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-industry.74. (cited on Page 70 and 88)

Bernhard Christoph, Britta Matthes, and Christian Ebner. Occupation-Based Measures—An Overview and Discussion. *KZfSS Kölner Zeitschrift für Soziologie und Sozialpsychologie*, 72(1):41–78, September 2020. ISSN 1861-891X. doi: 10.1007/s11577-020-00673-4. (cited on Page 2)

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What Does BERT Look at? An Analysis of BERT's Attention. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes, editors, *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. (cited on Page 18)

Jacob Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46, April 1960. ISSN 0013-1644. doi: 10.1177/001316446002000104. (cited on Page 85)

Frederick G. Conrad, Mick P. Couper, and Joseph W. Sakshaug. Classifying Open-Ended Reports: Factors Affecting the Reliability of Occupation Codes. *Journal of Official Statistics*, 32(1):75–92, March 2016. ISSN 2001-7367. doi: 10.1515/jos-2016-0003. (cited on Page 4)

Robert H. Creecy, Brij M. Masand, Stephen J. Smith, and David L. Waltz. Trading MIPS and Memory for Knowledge Engineering. *Communications of the ACM*, 35 (8):48–64, August 1992. ISSN 0001-0782, 1557-7317. doi: 10.1145/135226.135228. (cited on Page 5)

G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989. ISSN 0932-4194, 1435-568X. doi: 10.1007/BF02551274. (cited on Page 18)

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pages 16344–16359, Red Hook, NY, USA, November 2022. Curran Associates Inc. ISBN 978-1-7138-7108-8. (cited on Page 142)

Katharina Dengler and Britta Matthes. Digitalisierung in Deutschland: Substituierbarkeitspotenziale von Berufen und die Möglichen Folgen für die Beschäftigung. In Rolf Dobischat, Bernd Käpplinger, Gabriele Molzberger, and Dieter Münk, editors, *Bildung 2.1 Für Arbeit 4.0?*, pages 49–62. Springer Fachmedien Wiesbaden, Wiesbaden, 2019. ISBN 978-3-658-23373-0. doi: 10.1007/978-3-658-23373-0_3. (cited on Page 3)

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), NAACL'19*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. (cited on Page 5, 6, and 19)

Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Anh Tuan Luu, and Shafiq Joty. Data Augmentation Using Large Language Models: Data Perspectives, Learning Paradigms and Challenges. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the*

Association for Computational Linguistics: ACL 2024, pages 1679–1705, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.97. (cited on Page 7)

Ralf Dorau and Kristine Hein. Towards the Automated Classification of German Job Titles According to KldB. pages 681–686, October 2025. doi: 10.15439/2025F4081. (cited on Page 3)

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss Library. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2401.08281. (cited on Page 21)

Edmund Evangelista, Fathima Ruba, Salman Bukhari, Amril Nazir, and Ravishankar Sharma. GraphRAG-Enabled Local Large Language Model for Gestational Diabetes Mellitus: Development of a Proof-of-Concept. *JMIR Diabetes*, 11:e76454, January 2026. ISSN 2371-4379. doi: 10.2196/76454. (cited on Page 54)

Emily Eyles, David Manley, and Kelvyn Jones. Occupied with Classification: Which Occupational Classification Scheme Better Predicts Health Outcomes? *Social Science & Medicine*, 227:56–62, April 2019. ISSN 0277-9536. doi: 10.1016/j.socscimed.2018.09.020. (cited on Page 3)

Gregor Fabian, Choni Flöther, and Dirk Reifenberg, editors. *Generation Hochschulabschluss: Neue und Alte Differenzierungen: Ergebnisse des Absolventenpanels 2017*. Münster, 2021. ISBN 978-3-8309-9391-9. doi: 10.31244/9783830993919. (cited on Page 1 and 3)

Gregor Fabian, Choni Flöther, and Sarah Reifenberg. Daten- Und Methodenbericht Des Datenpakets "Absolventenbefragung 2017". Data and Methods Report, Deutsches Zentrum für Hochschul- und Wissenschaftsforschung (DZHW); Institut für angewandte Statistik (ISTAT); International Centre for Higher Education Research Kassel (INCHER-Kassel), Hannover, July 2023. (cited on Page 1, 3, and 28)

Andreas Fischer and Jens Dörpinghaus. Web Mining of Online Resources for German Labor Market Research and Education: Finding the Ground Truth? *Knowledge*, 4 (1):51–67, February 2024. ISSN 2673-9585. doi: 10.3390/knowledge4010003. (cited on Page 3)

Thibault Formal, C. Lassance, Benjamin Piwowarski, and S. Clinchant. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. *ArXiv*, September 2021a. doi: 10.48550/arXiv.2109.10086. (cited on Page 130)

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. *Proceedings of the 44th International ACM Conference on Research and Development in Information Retrieval, SIGIR'21*, pages 2288–2292, July 2021b. doi: 10.1145/3404835.3463098. (cited on Page 130)

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise Zero-Shot Dense Retrieval Without Relevance Labels. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.99. (cited on Page 7 and 24)

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP'21*, pages 6894–6910, 2021. doi: 10.18653/v1/2021.emnlp-main.552. (cited on Page 21)

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint*, March 2024. doi: 10.48550/arXiv.2312.10997. (cited on Page 23 and 24)

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer Feed-Forward Layers Are Key-Value Memories. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP'21*, pages 5484–5495, 2021. doi: 10.18653/v1/2021.emnlp-main.446. (cited on Page 18)

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. (cited on Page 18)

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang,

Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 Incentivizes Reasoning in LLMs Through Reinforcement Learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. (cited on Page 7, 19, and 88)

Hyukjun Gweon, Matthias Schonlau, Lars Kaczmirek, Michael Blohm, and Stefan Steiner. Three Methods for Occupation Coding Based on Statistical Learning. *Journal of Official Statistics*, 33(1):101–122, February 2017. doi: 10.1515/jos-2017-0006. (cited on Page 4 and 5)

Andreas M Hein. Identification and Bridging of Semantic Gaps in the Context of Multi-Domain Engineering. In *Forum on Philosophy, Engineering & Technology*, pages 58–57, January 2010. (cited on Page 3)

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv: Learning*, June 2016. doi: 10.48550/arXiv.1606.08415. (cited on Page 18)

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. (cited on Page 16)

Ute Hoffstätter, Miriam Kühne, and Thorben Sembritzki. DZHW-Absolventenpanel 2013. Daten- und Methodenbericht zur Absolvent*Innenkohorte 2013 (1. und 2. Befragungswelle). Data and Methods Report, Deutsches Zentrum für Hochschul- und Wissenschaftsforschung (DZHW), Hannover, October 2021. (cited on Page 1, 3, and 28)

Mengze Hong, Chen Jason Zhang, Di Jiang, Yuanfeng Song, Lu Wang, Yuanqin He, Zhiyang Su, and Qing Li. Expanding Chatbot Knowledge in Customer Service: Context-Aware Similar Question Generation Using Large Language Models. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2410.12444. (cited on Page 7)

K. Hornik, M. Stinchcombe, and H. White. Multilayer Feedforward Networks Are Universal Approximators. *Neural Netw.*, 2(5):359–366, July 1989. ISSN 0893-6080. (cited on Page 18)

Erik Johannes Husom, Arda Goknil, Merve Astekin, Lwin Khin Shar, Sagar Sen, Benedikt Andreas Mithassel, and Ahmet Soylu. Sustainable LLM Inference for Edge AI: Evaluating Quantized LLMs for Energy Efficiency, Output Accuracy, and Inference Latency. *ACM Transactions on Internet of Things*, 6(4):1–35, November 2025. ISSN 2691-1914, 2577-6207. doi: 10.1145/3767742. (cited on Page 7, 19, and 88)

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. Query Expansion by Prompting Large Language Models. *arXiv preprint*, 2023. doi: 10.48550/arXiv.2305.03653. (cited on Page 100, 103, and 106)

Kalervo Järvelin and Jaana Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 20(4):422–446, October 2002. ISSN 1046-8188. doi: 10.1145/582415.582418. (cited on Page 85)

Iman Johary, Raphael Romero, Alexandru C. Mara, and Tijl De Bie. JobHop: A Large-Scale Dataset of Career Trajectories. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2505.07653. (cited on Page 1)

Jeff Johnson, Matthijs Douze, and Herve Jegou. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, July 2021. ISSN 2332-7790, 2372-2096. doi: 10.1109/TBDATA.2019.2921572. (cited on Page 21)

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. Pearson Prentice Hall, 3rd edition, 2026. (cited on Page 15)

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP'20*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. (cited on Page 6, 21, 22, and 86)

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. *arXiv preprint*, October 2023. doi: 10.48550/arXiv.2310.03714. (cited on Page 143)

Minsang Kim, Cheoneum Park, and Seung Jun Baek. QPaug: Question and Passage Augmentation for Open-Domain Question Answering of LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics, EMNLP'24*, pages 9024–9042, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.527. (cited on Page 7)

Dorothea Kobler-Trill. *Das Kurzwort im Deutschen: Eine Untersuchung zu Definition, Typologie und Entwicklung*, volume 149 of *Reihe Germanistische Linguistik*. Max Niemeyer Verlag, Tübingen, 1994. (cited on Page 49)

Olga Kononykhina, Malte Schierholz, and Frauke Kreuter. The Impact of Question Framing on the Precision of Automatic Occupation Coding. *arXiv preprint*, January 2025. doi: 10.48550/arXiv.2501.05584. (cited on Page 4, 7, and 95)

Hamin Koo, Minseon Kim, and Sung Ju Hwang. Optimizing Query Generation for Enhanced Document Retrieval in RAG. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2407.12325. (cited on Page 7, 8, 24, 87, 100, and 103)

Mathijs A. Langezaal, Egon L. van den Broek, Susan Peters, Marcel Goldberg, Grégoire Rey, Melissa C. Friesen, Sarah J. Locke, Nathaniel Rothman, Qing Lan, and Roel C. H. Vermeulen. Artificial Intelligence Exceeds Humans in Epidemiological Job Coding. *Communications Medicine*, 3(1):160, November 2023. ISSN 2730-664X. doi: 10.1038/s43856-023-00397-4. (cited on Page 1 and 4)

Mathijs A. Langezaal, Egon L. van den Broek, Grégoire Rey, Nicole Le Moual, Corinne Pilorget, Marcel Goldberg, Roel Vermeulen, and Susan Peters. OPERAS Decision Support System versus Manual Job Coding: A Quantitative Analysis on Coding Time and Inter-Coder Reliability. *Occupational and Environmental Medicine*, June 2025. ISSN 1351-0711, 1470-7926. doi: 10.1136/oemed-2024-109823. (cited on Page 1 and 4)

Seokgi Lee. Transforming Questions and Documents for Semantically Aligned Retrieval-Augmented Generation. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2508.09755. (cited on Page 11 and 69)

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 9459–9474, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-7138-2954-6. (cited on Page 6 and 23)

Nan Li, Bo Kang, and Tijl De Bie. LLM4Jobs: Unsupervised Occupation Extraction and Standardization Leveraging Large Language Models. *Knowledge-Based Systems*, 316:113302, May 2025. ISSN 0950-7051. doi: 10.1016/j.knosys.2025.113302. (cited on Page 6 and 7)

Zhicheng Lin. Techniques for Supercharging Academic Writing with Generative AI. *Nature Biomedical Engineering*, 9(4):426–431, March 2024. ISSN 2157-846X. doi: 10.1038/s41551-024-01185-8. (cited on Page 138 and 143)

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query Rewriting in Retrieval-Augmented Large Language Models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP'23*, pages 5303–5315, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.322. (cited on Page 7, 8, 24, 87, 100, 103, and 106)

Yu A. Malkov and D. A. Yashunin. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 42(4):824–836, April 2020. doi: 10.1109/TPAMI.2018.2889473. (cited on Page 22)

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008. ISBN 978-0-521-86571-5. doi: 10.1017/CBO9780511809071. (cited on Page 85 and 95)

Lei Mao. Transformer Explained in One Single Page, June 2019. URL https://leimao.github.io/blog/Transformer-Explained/. Accessed: 2026-03-10. (cited on Page 20)

Ananda Martin-Caughey. What's in an Occupation? Investigating Within-Occupation Variation and Gender Segregation Using Job Titles and Task Descriptions. *American Sociological Review*, 86(5):960–999, October 2021. ISSN 0003-1224, 1939-8271. doi: 10.1177/00031224211042053. (cited on Page 4)

Tomas Mikolov, Kai Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, January 2013a. (cited on Page 21)

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff, editors, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL'13*, pages 746–751, Atlanta, Georgia, June 2013b. Association for Computational Linguistics. (cited on Page 16 and 17)

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey. *ACM Computing Surveys*, 56(2):1–40, September 2023. ISSN 0360-0300. doi: 10.1145/3605943. (cited on Page 5, 15, 16, and 19)

Mistral-AI, Abhinav Rastogi, Albert Q. Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, Léonard Blier, Lucile Saulnier, Matthieu Dinot, Maxime Darrin, Neha Gupta, Roman Soletskyi, Sagar Vaze, Teven Le Scao, Yihan Wang, Adam Yang, Alexander H. Liu, Alexandre Sablayrolles, Amélie Héliou, Amélie Martin, Andy Ehrenberg, Anmol Agarwal, Antoine Roux, Arthur Darcet, Arthur Mensch, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Chris Bamford, Christian Wallenwein, Christophe Renaudin, Clémence Lanfranchi, Darius Dabert, Devon Mizelle, Diego de las Casas, Elliot Chane-Sane, Emilien Fugier, Emma Bou Hanna, Gauthier Delerce, Gauthier Guinet, Georgii Novikov, Guillaume Martin, Himanshu Jaju, Jan Ludziejewski, Jean-Hadrien Chabran, Jean-Malo Delignon, Joachim Studnia, Jonas Amar, Josselin Somerville Roberts, Julien Denize, Karan Saxena, Kush Jain, Lingxiao Zhao, Louis Martin, Luyu Gao, Lélio Renard Lavaud, Marie Pellat, Mathilde Guillaumin, Mathis Felardos, Maximilian Augustin, Mickaël Seznec, Nikhil Raghuraman, Olivier Duchenne, Patricia Wang, Patrick von Platen, Patryk Saffer, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Romain Sauvestre, Rémi Delacourt, Sanchit Gandhi, Sandeep Subramanian, Shashwat

Dalal, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Thibault Schueller, Thibaut Lavril, Thomas Robert, Thomas Wang, Timothée Lacroix, Valeriia Nemychnikova, Victor Paltz, Virgile Richard, Wen-Ding Li, William Marshall, Xuanyu Zhang, and Yunhao Tang. Magistral, June 2025. (cited on Page 19)

Gerrit Müller. Panel Evidence on Within-Occupation Change in Job Tasks and Individual Wages. Working Paper 02/2024, IAB-Discussion Paper, 2024. (cited on Page 2)

Robert T. O'Reagan. Computer-Assigned Codes from Verbal Responses. *Communications of the ACM*, 15(6):455–459, June 1972. doi: 10.1145/361405.361419. (cited on Page 4)

Mehul D. Patel, Kathryn M. Rose, Cindy R. Owens, Heejung Bang, and Jay S. Kaufman. Performance of Automated and Manual Coding Systems for Occupational Data: A Case Study of Historical Records. *American Journal of Industrial Medicine*, 55(3):228–231, March 2012. ISSN 0271-3586. doi: 10.1002/ajim.22005. (cited on Page 1 and 4)

Avinash Patil and Aryan Jadon. Advancing Reasoning in Large Language Models: Promising Methods and Approaches. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2502.03671. (cited on Page 7 and 88)

Wiebke Paulus and Britta Matthes. The German Classification of Occupations 2010: Structure, Coding and Conversion Table. FDZ-Methodenreport 201308, Institut für Arbeitsmarkt- und Berufsforschung, Nürnberg, Germany, 2013. URL https://doku.iab.de/fdz/reporte/2013/MR_08-13_EN.pdf. (cited on Page 1, 2, 109, and 115)

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph Retrieval-Augmented Generation: A Survey. *ACM Transactions on Information Systems*, 44(2), December 2025. ISSN 1046-8188. doi: 10.1145/3777378. (cited on Page 54)

Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. Revisiting Demonstration Selection Strategies in In-Context Learning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9090–9101, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.492. (cited on Page 7)

Alec Radford and Karthik Narasimhan. Improving Language Understanding by Generative Pre-Training. Technical Report, OpenAI, 2018. (cited on Page 5, 16, and 19)

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural*

Language Processing, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. (cited on Page 19, 21, and 22)

Marvin Reuter, Mariann Rigó, Maren Formazin, Falk Liebers, Ute Latza, Stefanie Castell, Karl-Heinz Jöckel, Karin Halina Greiser, Karin B. Michels, Gérard Krause, Stefan Albrecht, Ilter Öztürk, Oliver Kuss, Klaus Berger, Benedikt M. J. Lampl, Michael Leitzmann, Hajo Zeeb, Karla Romero Starke, Sabine Schipf, Claudia Meinke-Franze, Wolfgang Ahrens, Andreas Seidler, Bianca Klee, Tobias Pischon, Andreas Deckert, Börge Schmidt, Rafael Mikolajczyk, André Karch, Barbara Bohn, Hermann Brenner, Bernd Holleczek, and Nico Dragano. Occupation and SARS-CoV-2 Infection Risk among 108 960 Workers during the First Pandemic Wave in Germany. *Scandinavian Journal of Work, Environment & Health*, 48(6): 446–456, September 2022. ISSN 1795-990X. doi: 10.5271/sjweh.4037. (cited on Page 3)

Katrin Rickmeier. Navigating Regional Barriers to Job Mobility: The Role of Opportunity Structures in Individual Job-to-Job Transitions. *Social Sciences*, 12 (5):295, May 2023. ISSN 2076-0760. doi: 10.3390/socsci12050295. (cited on Page 3)

Sidharth Rony and Jack Patman. Standard Occupation Classifier – A Natural Language Processing Approach. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2511.23057. (cited on Page 1)

Daniel E. Russ, Kwan-Yuet Ho, Joanne S. Colt, Karla R. Armenti, Dalsu Baris, Wong-Ho Chow, Faith Davis, Alison Johnson, Mark P. Purdue, Margaret R. Karagas, Kendra Schwartz, Molly Schwenn, Debra T. Silverman, Calvin A. Johnson, and Melissa C. Friesen. Computer-Based Coding of Free-Text Job Descriptions to Efficiently Identify Occupations in Epidemiological Studies. *Occupational and Environmental Medicine*, 73(6):417–424, June 2016. ISSN 1470-7926. doi: 10.1136/oemed-2015-103152. (cited on Page 1 and 4)

Parisa Safikhani, Hayastan Avetisyan, Dennis Föste-Eggers, and David Broneske. Automated Occupation Coding with Hierarchical Features: A Data-Centric Approach to Classification with Pre-Trained Language Models. *Discover Artificial Intelligence*, 3(1):6, February 2023. ISSN 2731-0809. doi: 10.1007/s44163-023-00050-y. (cited on Page 4, 5, 7, 28, 29, 30, 43, 94, 109, and 114)

Grant Sanderson. Neural Networks. YouTube Playlist, published by 3Blue1Brown, 2024. URL https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi. Accessed: 2026-03-11. Includes Chapter 5 (Transformers, The Tech Behind LLMs), Chapter 6 (Attention in Transformers, Step-by-Step), and Chapter 7 (How Might LLMs Store Facts). (cited on Page 16, 17, 18, and 19)

Malte Schierholz. Automating Survey Coding for Occupation. Master's thesis, Ludwig Maximilian University of Munich, 2014. URL https://epub.ub.uni-muenchen.de/21444/. (cited on Page 1, 3, and 4)

Malte Schierholz and Matthias Schonlau. Machine Learning for Occupation Coding – A Comparison Study. *Journal of Survey Statistics and Methodology, Oxford*

University Press, 9(5):1013–1034, November 2020. ISSN 2325-0984. doi: 10.1093/jssam/smaa023. (cited on Page 2, 4, 5, 7, and 109)

Malte Schierholz, Miriam Gensicke, Nikolai Tschersich, and Frauke Kreuter. Occupation Coding During the Interview. *Journal of the Royal Statistical Society Series A: Statistics in Society, John Wiley & Sons*, 181(2):379–407, February 2018. ISSN 0964-1998. doi: 10.1111/rssa.12297. (cited on Page 2, 3, 4, and 7)

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval Augmentation Reduces Hallucination in Conversation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.320. (cited on Page 6 and 23)

Marina Sokolova and Guy Lapalme. A Systematic Analysis of Performance Measures for Classification Tasks. *Information Processing & Management, Elsevier*, 45(4): 427–437, July 2009. ISSN 0306-4573. doi: 10.1016/j.ipm.2009.03.002. (cited on Page 85)

Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. Is Cosine-Similarity of Embeddings Really About Similarity? In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, pages 887–890, New York, NY, USA, May 2024. Association for Computing Machinery. ISBN 979-8-4007-0172-6. doi: 10.1145/3589335.3651526. (cited on Page 22)

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. Jina-Embeddings-v3: Multilingual Embeddings with Task LoRA. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2409.10173. (cited on Page 6, 7, and 22)

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One Embedder, Any Task: Instruction-Finetuned Text Embeddings. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL'2023*, pages 1102–1121, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.71. (cited on Page 7, 22, and 86)

Dewang Sultania, Zhaoyu Lu, Twisha Naik, Franck Dernoncourt, David Seunghyun Yoon, Sanat Sharma, Trung Bui, Ashok Gupta, Tushar Vatsa, Suhas Suresha, Ishita Verma, Vibha Belavadi, Cheng Chen, and Michael Friedrich. Domain-Specific Question Answering with Hybrid Search. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2412.03736. (cited on Page 130)

Bruno Taillé, Vincent Guigue, and Patrick Gallinari. Contextualized Embeddings in Named-Entity Recognition: An Empirical Study on Generalization. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II*, pages 383–391,

Berlin, Heidelberg, April 2020. Springer-Verlag. ISBN 978-3-030-45441-8. doi: 10.1007/978-3-030-45442-5_48. (cited on Page 19)

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What Do You Learn from Context? Probing for Sentence Structure in Contextualized Word Representations. *ArXiv*, May 2019. doi: 10.48550/arXiv.1905.06316. (cited on Page 18)

Nandan Thakur, Nils Reimers, Andreas Ruckl'e, Abhishek Srivastava, and Iryna Gurevych. BEIR: A Heterogenous Benchmark for Zero-Shot Evaluation of Information Retrieval Models. *ArXiv*, April 2021. doi: 10.48550/arXiv.2104.08663. (cited on Page 11 and 91)

Kea Tijdens. Dropout Rates and Response Times of an Occupation Search Tree in a Web Survey. *Journal of Official Statistics, SAGE Publications*, 30(1):23–43, March 2014. ISSN 0282-423X. doi: 10.2478/jos-2014-0002. (cited on Page 4)

Teemu Vahtola, Mathias Creutz, and Jörg Tiedemann. It Is Not Easy to Detect Paraphrases: Analysing Semantic Similarity with Antonyms and Negation Using the New SemAntoNeg Benchmark. In Jasmijn Bastings, Yonatan Belinkov, Yanai Elazar, Dieuwke Hupkes, Naomi Saphra, and Sarah Wiegreffe, editors, *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 249–262, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.blackboxnlp-1.20. (cited on Page 21)

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. (cited on Page 5, 16, 17, 18, 19, and 20)

Anton Voronov, Lena Wolf, and Max Ryabinin. Mind Your Format: Towards Consistent Evaluation of In-Context Learning Improvements. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6287–6310, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.375. (cited on Page 7)

Wenxin Wan, Calvin B Ge, Melissa C Friesen, Sarah J Locke, Daniel E Russ, Igor Burstyn, Christopher J O Baker, Anil Adisesh, Qing Lan, Nathaniel Rothman, Anke Huss, Martie van Tongeren, Roel Vermeulen, and Susan Peters. Automated Coding of Job Descriptions From a General Population Study: Overview of Existing Tools, Their Application and Comparison. *Annals of Work Exposures and Health*, 67(5):663–672, February 2023. ISSN 2398-7308. doi: 10.1093/annweh/wxad002. (cited on Page 4)

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query Expansion with Large Language Models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language*

Processing, EMNLP'23, pages 9414–9423, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.585. (cited on Page 7)

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving Text Embeddings with Large Language Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.642. (cited on Page 22)

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual E5 Text Embeddings: A Technical Report. *arXiv preprint*, February 2024b. doi: 10.48550/arXiv.2402.05672. (cited on Page 6 and 22)

Zaitian Wang, Jinghan Zhang, Xinhao Zhang, Kunpeng Liu, Pengfei Wang, and Yuanchun Zhou. Diversity-Oriented Data Augmentation with Large Language Models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22265–22283, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1084. (cited on Page 134)

Anja Warning, Enzo Weber, and Anouk Püffel. On the Impact of Digitalization and Artificial Intelligence on Employers' Flexibility Requirements in Occupations—Empirical Evidence for Germany. *Frontiers in Artificial Intelligence*, 5, May 2022. ISSN 2624-8212. doi: 10.3389/frai.2022.868789. (cited on Page 3)

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent Abilities of Large Language Models. *arXiv preprint*, 2022. doi: 10.48550/arXiv.2206.07682. (cited on Page 19)

Mingrui Wu and Sheng Cao. LLM-Augmented Retrieval: Enhancing Retrieval Models Through Language Models and Doc-Level Embedding. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2404.05825. (cited on Page 7)

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective Long-Context Scaling of Foundation Models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4643–4663, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.260. (cited on Page 7)

Chejian Xu, Wei Ping, Peng Xu, Zihan Liu, Boxin Wang, Mohammad Shoeybi, Bo Li, and Bryan Catanzaro. From 128K to 4M: Efficient Training of Ultra-Long Context Large Language Models. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2504.06214. (cited on Page 7)

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2505.09388. (cited on Page 7, 19, 88, and 100)

Danyang Zhang, Junhao Song, Ziqian Bi, Xinyuan Song, Yingfang Yuan, Tianyang Wang, Joe Yeong, and Junfeng Hao. Mixture of Experts in Large Language Models. *arXiv preprint*, 2025a. doi: 10.48550/arXiv.2507.11181. (cited on Page 7, 19, and 100)

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint*, 2025b. doi: 10.48550/arXiv.2506.05176. (cited on Page 7 and 22)

I herewith assure that I have written the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or according to their meaning.

I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Furthermore, I confirm that I am aware that the use of content (including but not limited to text, figures, images and code) generated by AI in the thesis must be disclosed. In those cases I have specified the AI system used, I have marked the specific sections of the thesis where AI-generated content was used, and I have provided a brief explanation of the level of detail at which the AI system was used to generate the content. I also stated the reason for using the tools.

I assure that even if a generative AI system has been used, the scientific contribution has been made entirely by myself.

Andy Koch

Magdeburg, March 21, 2026