SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Towards Scalable Domain-Specific Document Annotation: A Semantic Archetype-Driven Framework

Moritz Steigerwald

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Towards Scalable Domain-Specific Document Annotation: A Semantic Archetype-Driven Framework

Skalierbare domänenspezifische Dokumentannotierung: Ein semantisches Archetyp-gestütztes Framework

Author:	Moritz Steigerwald
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Stephen Meisenbacher
Submission Date:	1st October 2025

I confirm that this master's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 30.05.2025

Location, Submission Date

Author

AI Assistant Usage Disclosure

Introduction

Performing work or conducting research at the Chair of Software Engineering for Business Information Systems (sebis) at TUM often entails dynamic and multi-faceted tasks. At sebis, we promote the responsible use of *AI Assistants* in the effective and efficient completion of such work. However, in the spirit of ethical and transparent research, we require all student researchers working with sebis to disclose their usage of such assistants.

For examples of correct and incorrect AI Assistant usage, please refer to the original, unabridged version of this form, located at this link.

Use of *AI Assistants* for Research Purposes

I have used AI Assistant(s) for the purposes of my research as part of this thesis.

X Yes ☐ No ☐

Explanation:

- **Language support and writing assistance**: A large language model was used to enhance clarity, grammar, phrasing, and tone in the thesis writing. All text was reviewed and adapted by the author.
- **Literature discovery and research navigation**: The assistant was used to brainstorm search terms, refine queries, and surface candidate papers, venues, and keywords. All citations in this thesis originate from primary sources that were accessed and read by the author. AI outputs were not cited as sources.
- **Editing and formatting support**: The assistant provided suggestions for LaTeX usage, reference styling, table and figure caption wording, and minor structural edits. All formatting changes were checked by the author.

I confirm in signing below, that I have reported all usage of AI Assistants for my research, and that the report is truthful and complete.

Munich, 30.05.2025

Location, Date

Author

Abstract

The growing demand for Natural Language Processing (NLP) in specialized, domain-specific contexts, where large annotated corpora are scarce, creates the need for frameworks that transform unstructured text into annotated datasets with minimal manual effort. While large language models offer zero-shot capabilities and weak supervision reduces labeling needs, both suffer from hallucinations, domain-specific inaccuracies, high costs, and noisy predictions that hinder practical use. This thesis addresses this gap by completing the *CreateData4AI (CD4AI)* pipeline with a *semantic archetype-driven framework* for efficient and accurate text annotation and classification.

Semantic archetypes, interpretable prototypes distilled from domain corpora, serve as the foundation of the framework. We implement four complementary methods: (1) embedding-based similarity classification as a baseline; (2) contrastive learning to refine domain-specific embeddings; (3) progressive pseudo-labeling that combines archetypes with minimal gold labels to iteratively expand training data; and (4) Group Relative Policy Optimization (GRPO), a reinforcement learning approach that filters noisy archetypes and optimizes subset selection for robustness.

Experiments on five heterogeneous datasets (20 Newsgroups, AG News, BBC News, DBpedia, arXiv) show that the framework achieves 83.5% average accuracy with only 25 labeled examples per class. We achieve 96.6% of fully supervised performance, while requiring 100× fewer labels. It outperforms traditional weak supervision baselines in both accuracy and efficiency (275.6 vs. 6.9 samples/s) and matches zero-shot LLM accuracy at 14× higher throughput without API dependencies.

These results underline three core strengths: interpretability through human-readable archetypes, scalability by reducing annotation needs by two orders of magnitude, and adaptability across domains without architectural changes. By completing the CD4AI pipeline, this work provides an end-to-end solution for automated annotation, enabling organizations to deploy accurate text classifiers within days rather than months and lowering the barrier to NLP adoption in specialized fields.

Contents

1 Introduction

The rapid advancement of artificial intelligence over the past decade has been driven by an unprecedented convergence of three fundamental forces: exponentially increasing computational power, vast quantities of digital data, and sophisticated learning algorithms. Although the first two have scaled dramatically, a critical bottleneck persists: human annotation of training data. This thesis addresses this fundamental challenge by developing and evaluating classification methods that complete the CreateData4AI pipeline for scalable domain-specific document annotation.

1.1 Motivation and Problem Statement

For nearly five decades, Moore's Law reliably predicted the doubling of transistor density every two years, driving steady improvements in computational capability [1]. As we approach the physical limits of silicon semiconductors, this growth has plateaued. In its place, massively parallel Graphics Processing Unit (GPU) architectures have emerged, sustaining and even accelerating compute growth for machine learning workloads [2]. Modern large language models (LLMs) such as OpenAI's GPT-4 [3] and Anthropic's Claude-3 [4] now require hundreds of thousands of GPU hours for pretraining. Parallel to this computational expansion, text data has proliferated across all domains. This abundance presents both opportunity and challenge: valuable knowledge exists within, but extracting it requires sophisticated natural language processing.

Despite these advances, a fundamental bottleneck remains: creating high-quality labeled datasets. While computational power scales exponentially and data grows geometrically, human annotation capacity remains stubbornly linear. This creates what A. Ratner, Bach, Ehrenberg, et al. [5] describes as the "key bottleneck in the deployment of machine learning systems."

The annotation problem manifests itself across multiple dimensions. Professional annotation services are costly and time-intensive, with substantial variability in cost, throughput, and quality across tasks [6]. Quality suffers too: interannotator agreement rarely exceeds 0.8 and often falls below 0.6 for nuanced tasks [7]. Specialized domains that require expert knowledge face even greater challenges.

The machine learning community has developed several mitigation strategies, each with limitations. Weak supervision frameworks such as Snorkel [5] generate labels programmatically, but require significant expertise. Active learning reduces requirements but keeps humans in the loop [8]. Transfer learning benefits from pre-trained representations but still often requires substantial in-domain labeled data, especially under distribution shift [9].

Few-shot learning [10] shows promise but lags behind supervised methods.

While LLMs offer automated annotation at scale, they introduce new challenges: *hallucination* in specialized domains [11], redundant annotations that miss data diversity, and the fundamental evaluation paradox, assessing quality requires the very ground truth we seek to avoid creating.

This thesis proposes a novel solution to the annotation bottleneck through the concept of *semantic archetypes*; interpretable, representative patterns that capture the essential characteristics of document classes within a specific domain. Rather than annotating individual documents, we focus on discovering and refining a small set of archetypes that can then guide the automated annotation of entire corpora. The approach builds upon the CreateData4AI (CD4AI) pipeline, a comprehensive framework for automated dataset creation that addresses the full lifecycle from raw text to annotated corpora.

Our framework operates on several key principles. Instead of labeling thousands of individual documents, we identify semantic archetypes that encapsulate class characteristics. These archetypes serve as "prototypes" that can be reviewed and validated by domain experts much more efficiently than examining entire datasets. Archetypes enable the generation of multiple *weak supervision signals*: similarity matching, keyword extraction, and pattern recognition; that can be combined to produce high-confidence pseudo-labels for unlabeled documents. Through iterative cycles of pseudo-labeling [12], confidence thresholding, and model training, our framework progressively expands the labeled dataset while maintaining quality. Unlike generic pre-trained models, our archetype-based approach naturally adapts to domain-specific patterns and terminology, as the archetypes themselves emerge from the target domain corpus.

This work completes the CreateData4AI pipeline, whose ultimate goal is fully automated corpus annotation with minimal human supervision. While CD4AI's existing components focus on *archetype distillation*, which extracts semantic patterns from unlabeled text using domain expert seed keywords, the pipeline lacked the final step: a classifier to annotate entire corpora using these archetypes. This thesis contributes that missing component, developing and evaluating classification methods that transform distilled archetypes into scalable annotation systems.

We develop and evaluate four complementary methods with increasing sophistication: (1) direct embedding-based archetype matching computes similarity between documents and archetypes, (2) contrastive learning [13] adapts embeddings to domain-specific characteristics, (3) progressive pseudo-labeling synthesizes multiple weak supervision signals to iteratively expand training sets, and (4) reinforcement learning [14] optimizes archetype selection through competitive policy training.

The ability to automatically annotate domain-specific corpora would transform numerous fields: scientific literature tracking, legal document categorization, healthcare record analysis, business intelligence, and educational content organization. Yet challenges remain: maintaining accuracy, mitigating bias, ensuring interpretability, and adapting to domain shift [15].

Our evaluation across five diverse datasets (20 Newsgroups [16], AG News [17], BBC

News [18], DBpedia [19], and arXiv [20]) demonstrates that these challenges can be successfully addressed. Using only 25 labeled examples per class combined with distilled semantic archetypes, our progressive pseudo-labeling method achieves 83.5% average accuracy, reaching 96.6% of fully supervised performance while reducing annotation requirements by two orders of magnitude. The approach processes 275.6 samples per second, enabling real-time deployment while operating 40× faster than graph-based weak supervision and 14× faster than zero-shot large language models at comparable accuracy. On well-structured datasets like BBC News and DBpedia, performance reaches 95–97%, while more challenging domains like arXiv achieve 75% accuracy. This combination of high accuracy, practical efficiency, and minimal supervision requirements validates the core premise: semantic archetypes can serve as effective bridges between unsupervised pattern discovery and supervised classification, enabling rapid deployment of domain-specific NLP systems without prohibitive annotation costs.

The remainder is organized as follows: Chapter 2 provides theoretical foundations. Chapter 3 presents the CreateData4AI pipeline and related work. Chapter 4 details our methodology. Chapter 5 presents evaluation and analysis. Chapter 6 discusses implications and limitations. Chapter 7 concludes.

Through this work, we demonstrate that the annotation bottleneck can be substantially alleviated through intelligent use of semantic archetypes, enabling scalable document annotation that keeps pace with exponential growth in compute and data.

1.2 Important Definitions and Terminology

Term	Definition
Seed Keywords	Initial 3–5 representative terms per class provided by domain expert
Keyword Expansion	Embedding-based growth of vocabulary to ∼30–50 terms per class
Context Window	Dependency-guided text snippet (3–10 tokens) around keywords
Hierarchical Clustering	Ward-linkage clustering [21] with recursive splits for semantic grouping
Semantic Archetype	Concise textual prototype with examples capturing cluster meaning
Archetype Distillation	LLM-based conversion of clusters into interpretable archetypes
Embedding Classification	Document classification via similarity to archetype embeddings
Progressive Pseudo-Labeling	Iterative training set expansion using high-confidence predictions
Weak Supervision	Combining multiple noisy signals for robust pseudo-labels [22]
GRPO	Group Relative Policy Optimization for archetype selection [23]

1.3 Research Questions and Contributions

This thesis addresses the fundamental challenge of scalable document annotation by completing the CreateData4AI pipeline with classification methods that leverage semantic archetypes. We investigate three core research questions:

1. **How can semantic archetypes be leveraged to classify and annotate domain-specific documents?**
   We explore how interpretable semantic patterns extracted through the CD4AI pipeline can serve as the foundation for document classification, examining the effectiveness of archetype-based weak supervision signals across diverse domains.

2. **Can a reward-based feedback system boost both classification accuracy and archetype quality?**
   We investigate whether reinforcement learning approaches, specifically our proposed GRPO method, can optimize archetype selection through competitive policy training, improving both the quality of selected archetypes and resulting classification performance when combined with gold examples.

3. **How does this archetype-driven framework compare to supervised models and zero-shot LLMs in terms of accuracy and resource utilization across different domains?** We conduct comprehensive evaluations comparing our minimal-supervision approach (requiring only 25 examples per class) against fully supervised baselines and state-of-the-art language models, measuring both classification accuracy and computational efficiency.

In addressing these questions, this thesis makes the following contributions:

- **Completion of the CreateData4AI Pipeline:** We develop and implement the final classification component that transforms CD4AI's distilled archetypes into a fully functional annotation system, enabling end-to-end automated corpus annotation from seed keywords to labeled datasets.
- **Four Complementary Classification Methods:** We present a progression of increasingly sophisticated approaches: (1) direct embedding-based archetype matching for baseline similarity classification, (2) contrastive learning for domain-specific embedding adaptation, (3) progressive pseudo-labeling that synthesizes multiple weak supervision signals for iterative training set expansion, and (4) GRPO reinforcement learning for optimal archetype subset selection.
- **Low-Resource Document Annotation Framework:** We demonstrate that competitive classification performance (achieving 83.5% average accuracy) is attainable with minimal labeled data; just 25 examples per class combined with semantic archetypes, representing a 100$\times$ reduction compared to traditional supervised approaches while maintaining interpretability through archetype-based weak supervision.
- **Comprehensive Multi-Domain Evaluation:** We validate our framework across five diverse datasets spanning news categorization, topic classification, and ontology classification, demonstrating robust performance and scalability from datasets with 4 classes to those with 20 classes, and from hundreds to thousands of archetypes per domain.
- **Open-Source Implementation:** We provide a complete, production-ready implementation integrated within the CD4AI framework, including all classification methods, evaluation scripts, and documentation, enabling immediate application to new domains and facilitating reproducible research.

Through these contributions, we demonstrate that the annotation bottleneck constraining machine learning deployment can be substantially alleviated through intelligent use of semantic archetypes, providing a practical path toward scalable, efficient, and interpretable document annotation systems that adapt naturally to domain-specific requirements while requiring minimal human supervision.

2 Theoretical Foundations

2.1 Text Embeddings and Vector Spaces

Natural language data in its raw form (words, sentences, documents) is inherently symbolic and discrete. To enable mathematical modeling and machine learning, we must *represent text in a continuous vector space*. In a vector space representation, linguistic units such as words or documents are mapped to geometric vectors, allowing us to quantify similarity, perform algebraic operations, and apply linear models. This idea has deep theoretical roots: *the distributional hypothesis* in linguistics states that words appearing in similar contexts tend to have similar meanings [24]. By representing words as vectors, we can utilize this hypothesis: words with similar distributions in text will end up nearby in the vector space, reflecting semantic resemblance. Embedding text into vectors is foundational for nearly all modern *Natural Language Processing (NLP)* pipelines. It bridges human language and quantitative algorithms: classification, clustering, information retrieval, and deep neural networks require numeric input features. Vector representations make it possible to compute distances or angles between texts, enabling tasks like search ranking and document similarity to be solved by well-defined mathematical measures (e.g., cosine similarity) and further assisting in more complex downstream tasks like classification and generation [25, pp. 54–60].

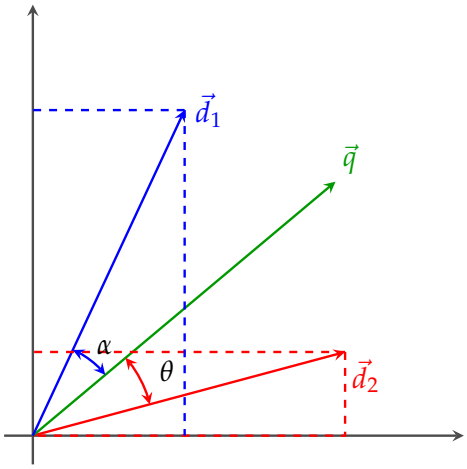

Figure 2.1: Illustration of the Vector-Space Model: two document vectors ($\vec{d}_1$ and $\vec{d}_2$) and a query vector ($\vec{q}$) are represented in a common term space. The angle α between $\vec{d}_1$ and $\vec{q}$ is smaller than the angle θ between $\vec{d}_2$ and $\vec{q}$, indicating that document 1 is more relevant to the query.[1]

In summary, *vector space models of text* offer a powerful abstraction: they encode textual items (from words to whole documents) as points in $\mathbb{R}^n$ (illustrated in Figure 2.1), such that proximity in this space correlates with linguistic or conceptual similarity [26, pp. 335–362]. This concept, introduced in information retrieval in the 1970s, was pioneered by Salton's SMART system, being one of the first works to represent documents as vectors of term weights [27]. It laid the groundwork for a wealth of text representation techniques that followed, from simple frequency-based vectors to advanced neural embeddings.

2.1.1 Bag-of-Words Model and tf-idf: Sparse Representations

The *Bag-of-Words (BOW)* model represents documents as high-dimensional sparse vectors where each dimension corresponds to a vocabulary term and values indicate term frequency. Despite ignoring word order, BOW vectors capture topical content through term overlap [27].

Term frequency-inverse document frequency (tf-idf) refines BOW by weighting terms according to their discriminative power:

$$\text{tfidf}(t, d, \mathcal{D}) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \times \log\left(\frac{|\mathcal{D}|}{1 + |\{d' \in \mathcal{D} \mid t \in d'\}|}\right), \tag{2.1}$$

where $f_{t,d}$ is the count of term t in document d, and the logarithmic factor penalizes terms that appear in many documents [28]. Terms that are frequent in a document but rare across the corpus receive high weights, making them salient for retrieval and classification. Combined with cosine similarity, tf-idf vectors underpinned decades of text classification research [29, 30]. However, sparse representations ignore semantic relationships and word order, limitations addressed by neural embeddings. Dimensionality reduction via Singular Value Decomposition (LSA) created dense low-dimensional representations that captured latent semantic factors [31], while probabilistic topic models like LDA yielded similar distributional embeddings [32].

2.1.2 Neural Word Embeddings: Word2Vec and GloVe

Neural language models learn distributed word representations by embedding words into continuous vector spaces [33]. *Word2Vec's* Skip-Gram and CBOW architectures [34, 35] efficiently train on large corpora, producing 100-300 dimensional vectors that encode rich semantic relationships through simple vector arithmetic:

$$\mathbf{v}_{\texttt{king}} - \mathbf{v}_{\texttt{man}} + \mathbf{v}_{\texttt{woman}} \approx \mathbf{v}_{\texttt{queen}}, \quad \mathbf{v}_{\texttt{France}} - \mathbf{v}_{\texttt{Paris}} + \mathbf{v}_{\texttt{Italy}} \approx \mathbf{v}_{\texttt{Rome}}.$$

GloVe [36] combines global matrix factorization with local context windows, learning vectors where co-occurrence statistics are preserved in the embedding space. Both methods revolutionized NLP by enabling pre-trained embeddings that substantially improve downstream tasks such as named entity recognition and sentiment classification [37, 38].

However, static embeddings assign one vector per word type, ignoring polysemy (e.g., `bank` as financial institution vs. riverbank). Contextual models address this limitation.

[1] Adapted from `https://en.wikipedia.org/wiki/Vector_space_model`

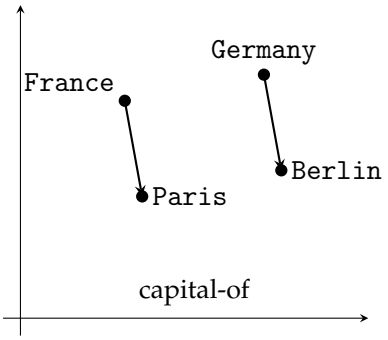

Figure 2.2: Country-capital offsets share the same direction in a 2-D projection of word embeddings.

2.1.3 Contextual Embeddings: Transformer-Based Language Models

Static vectors assign one embedding to each type of word, ignoring context. Modern models instead learn a *contextual* function.

$$f(\text{word}, \text{context}) \; : \; \mathbb{R}^d,$$

so that the vector for `bank` in a financial sentence differs substantially from `bank` beside a river. The first widely adopted contextual representations were obtained from a deep bidirectional language model, known as *ELMo*, which encoded tokens with stacked LSTMs and improved six benchmark tasks simply by concatenation to existing features [39]. Transformer encoders made this process faster and more expressive: the architecture's self-attention maps each token to all others without recurrence, capturing long-range dependencies in a single layer [40]. A bidirectionally masked Transformer yields *BERT*, whose top-layer token vectors pushed state-of-the-art performance in question answering, entailment, and nine other GLUE tasks [41]. Autoregressive pretraining with a unidirectional transformer produces the *GPT* family; even without supervised fine-tuning, these models transfer to diverse zero-shot settings [42]. Fine-tuning a pre-trained language model on downstream data -pioneered for classification in *ULMFiT* - shows that only a few hundred labeled examples can suffice when strong contextual embeddings are available [43]. Probing studies reveal that intermediate layers of such models encode part-of-speech tags, syntactic trees, and semantic roles, explaining their effectiveness across tasks [44, 45]. Training refinements such as larger corpora, dynamic masking, and longer sequences further strengthen performance, as demonstrated by *RoBERTa* [46].

Replacing static word2vec/GloVe features with these context-sensitive vectors consistently boosts accuracy in sequence labelling, question answering, sentiment analysis and more [39, 41]. Because the same pre-trained model can be fine-tuned for virtually any NLP task, contextual Transformer embeddings now serve as the default representation layer in modern language technology.

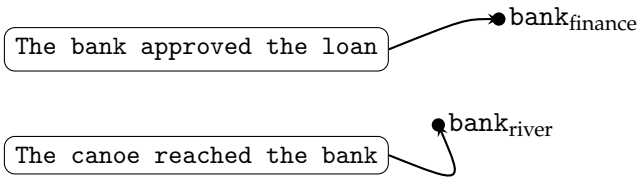

Figure 2.3: Contextual model assigns different vectors to the homograph `bank` depending on its usage.

2.2 Distance Metrics and Prototype-Based Classification

Vector representations by themselves do not yield actionable insight-one must still decide whether two texts are "close" or "far" in the embedding space. Section 2.2 therefore illustrates (1) the distance and similarity functions that underpin most text-vector operations, (2) how high-dimensional geometry affects their behaviour, and (3) how prototype-based classifiers such as the Rocchio algorithm exploit these measures. We begin with a concise catalogue of the most common metrics for text data.

2.2.1 Distance & Similarity Functions for Text

Vector representations require distance metrics to quantify similarity. The **Minkowski** (ℓ_p) family generalizes common metrics:

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}, \qquad p \geq 1,$$

with p=1 (Manhattan), p=2 (Euclidean), and $p \to \infty$ (Chebyshev) as special cases [47]. **Euclidean distance** measures straight-line separation and underlies k-means and nearest-centroid classifiers [30].

In high dimensions these ℓ_p metrics suffer from *distance concentration*: the ratio between nearest and farthest neighbours approaches unity, making rank-ordering unreliable [48, 49]. A related phenomenon, *hubness*, causes certain points to appear in many nearest-neighbour lists, distorting similarity search [50].

For sparse tf-idf vectors and dense contextual embeddings, comparing *angles* instead of lengths provides a remedy. **Cosine similarity**

$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$$

normalizes each vector onto the unit sphere, capturing direction rather than magnitude and remaining discriminative under high dimensionality [51, 29]. Cosine is therefore the de-facto choice in vector-space retrieval and serves as our primary similarity measure for archetype-based classification. Converting to a distance is straightforward: $d_{\cos} = 1 - \cos(\theta)$ obeys the metric axioms for non-negative vectors [49].

Alternative metrics include **Jaccard distance** $d_J(A, B) = 1 - |A \cap B|/|A \cup B|$ for binary term-presence vectors [52], and **Mahalanobis distance** that adapts Euclidean distance with a learned covariance matrix to whiten correlations [53].

Figure 2.4 illustrates distance concentration for Gaussian samples: the relative contrast $(D_{\max} - D_{\min})/D_{\min}$ converges rapidly towards 0 as dimension grows.

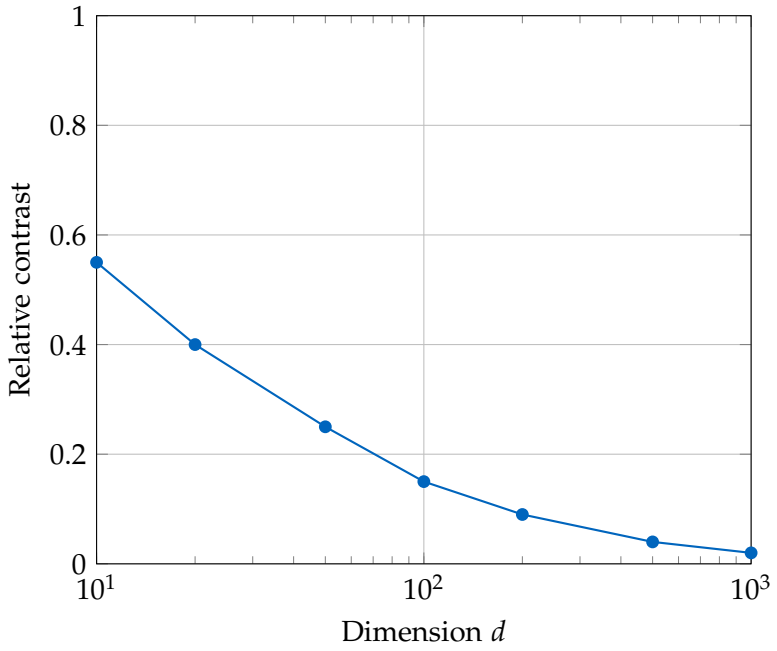

Figure 2.4: Distance concentration for synthetic Gaussian data: the relative spread between farthest and nearest neighbours shrinks as dimensionality increases.

Standard practice for text embeddings is to ℓ_2-normalize every vector and compute cosine similarity, which mitigates distance concentration while preserving relative neighborhood structure [54]. This motivates the prototype-based classifiers examined next.

2.2.2 Prototype-Based Classification: Rocchio & Nearest-Centroid

Prototype methods summarise each class by one (or a few) exemplar vectors, reducing classification to a single distance computation [30]. Their simplicity is attractive for high-dimensional text data, where storing or searching all training points is expensive.

Nearest-centroid rule. Given labelled documents $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ represented in a vector space, compute the *class centroid*

$$\boldsymbol{\mu}_c = \frac{1}{|D_c|} \sum_{i\,:\,y_i = c} \mathbf{x}_i, \tag{2.2}$$

for each class c. A test vector $\mathbf{x}$ is then assigned

$$\hat{y} \;=\; \arg\min_{c} \; d\big(\mathbf{x},\, \boldsymbol{\mu}_c\big), \tag{2.3}$$

where d is usually cosine distance for tf-idf inputs [51]. Equation (2.3) yields linear decision boundaries orthogonal to the line joining competing centroids (see Figure 2.5) [49].

Rocchio classifier. The *Rocchio algorithm* originated as a relevance-feedback mechanism in the SMART system [55]. For binary text categorisation it constructs a prototype

$$\mathbf{p} \;=\; \alpha\,\frac{1}{|D^{+}|}\sum_{\mathbf{d}\in D^{+}}\mathbf{d} \;-\; \beta\,\frac{1}{|D^{-}|}\sum_{\mathbf{d}\in D^{-}}\mathbf{d}, \tag{2.4}$$

where D^{+} and D^{-} are the sets of positive and negative training documents, and $\alpha, \beta > 0$ weight their influence. A test document is relevant if the cosine between $\mathbf{p}$ and $\mathbf{x}$ exceeds a threshold [55]. Empirical studies show Rocchio rivals k-NN on news filtering while requiring orders-of-magnitude fewer comparisons [56, 30].

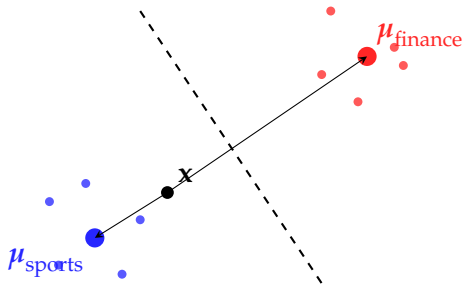

Figure 2.5: Nearest-centroid classification with cosine distance. The test document $\mathbf{x}$ falls on the *sports* side of the perpendicular bisector and is labelled accordingly.

Prototype classifiers are fast, memory-light and easily interpreted as topic "exemplars" [30]. They cope well with high-dimensional sparsity because centroids average away noise, and distance is computed only once per class. Yet they assume each class forms a roughly convex cluster; overlapping or multimodal categories degrade accuracy [30]. Hubness may still arise, but less severely because hubs are *averaged* into centroids [50].

While prototype methods rely on fixed distance functions, modern NLP systems usually *learn* a task-specific decision boundary by fine-tuning large Transformer encoders; this paradigm is the focus of the next section.

2.3 Supervised Transformer Models for Text Classification

The successes of contextual embeddings (2.1.3) derive from pre-training very deep Transformer encoders on unlabelled text and then adapting the same parameters to a supervised task with

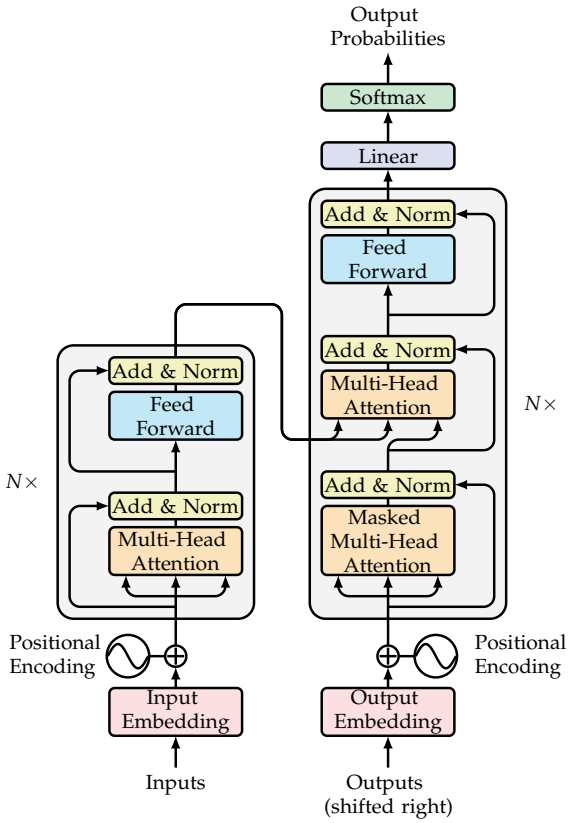

Figure 2.6: Schematic of a single Transformer encoder block.

only a small task-specific head. This section reviews the Transformer block (2.3.1), surveys the encoder variants later used in our experiments (2.3.2), and summarises the fine-tuning techniques that turn a general language model into a high-accuracy classifier (2.3.3).

2.3.1 Transformer Architecture and Pre-training Paradigm

The Transformer [40] replaces recurrence with *self-attention*, enabling parallel processing of sequences. Each layer computes scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)V,$$

where queries Q, keys K, and values V are linear projections of the input. *Multi-head attention* runs h such operations in parallel with different learned projections, concatenating outputs before a position-wise feed-forward network. Residual connections and layer normalization stabilize training of deep stacks (12-24 layers).

The original architecture (Figure 2.6) comprises an encoder stack (bidirectional context) and decoder stack (autoregressive generation). For text classification, we use **encoder-only**

Adapted from Vaswani *et al.* [40]; TikZ template by Negrinho (`https://github.com/negrinho/sane_tikz`).

models: the encoder produces contextual token representations, and the final-layer `[CLS]` token embedding serves as the document representation [41].

Pre-training paradigms. **Masked Language Modeling (MLM)** randomly masks 15% of input tokens and trains the model to predict them using bidirectional context, yielding encoders like BERT [41]. **Causal Language Modeling (CLM)** predicts the next token autoregressively, producing decoders like GPT [42]. **Replaced-token detection** (ELECTRA) trains a discriminator to identify corrupted tokens, providing more efficient learning [57].

After pre-training on large unlabeled corpora, these models are fine-tuned on labeled tasks by adding a classification head on `[CLS]` and updating all parameters with a low learning rate. This transfer learning paradigm has become standard for NLP [41, 46]. The next subsection details specific encoder variants used in our experiments.

2.3.2 Encoder Variants for Classification: BERT, RoBERTa, ELECTRA, DeBERTa

Transformer encoders differ mainly in their *pre-training objective* and the amount or quality of data they see. Table 4.4 highlights four variants we employ later; the remainder of this subsection summarises their design choices and empirical impact.

BERT. The baseline encoder uses masked-language modelling (MLM) and next-sentence prediction (NSP) on BookCorpus & English Wikipedia [41]. A single `[CLS]` token is prepended; its final hidden state becomes a task vector that a soft-max layer maps to class labels. Fine-tuning BERT yields large gains on the GLUE benchmark [58].

RoBERTa. Y. Liu, Ott, Goyal, et al. [46] show that BERT was *under-trained*. By removing NSP, switching to dynamic masking, training for more steps on a 160 GB corpus and using larger mini-batches, RoBERTa exceeds BERT by +3-5 GLUE points while keeping the architecture unchanged [46].

ELECTRA. MLM wastes 85 contribute to the loss. ELECTRA replaces MLM with *replaced-token detection*: a small generator corrupts the input and a discriminator predicts whether each token is original or fake [57]. This yields BERT-level accuracy with $\approx$25 % of the compute [57].

DeBERTa. DeBERTa disentangles *content* and *position* vectors in attention, letting the model learn relative positions explicitly, and it introduces an enhanced mask decoder [59]. With the same data budget, DeBERTa beats RoBERTa on GLUE and pushes a single model above human performance on SuperGLUE [59].

Taken together, these variants illustrate a trend: architectural tweaks are minor, while changes in the *training signal* and *data regime* account for most performance gains. The next subsection (2.3.3) shows how task-specific fine-tuning strategies unlock this potential.

Table 2.1: Key differences among encoder variants. All use the same 12-layer base architecture.

Model	Pre-training objective	Extra data	Reported GLUE ↑
$\text{BERT}_{\text{base}}$	MLM + NSP	16 GB	79.6 [41]
$\text{RoBERTa}_{\text{base}}$	MLM (no NSP, dynamic)	160 GB	83.0 [46]
$\text{ELECTRA}_{\text{base}}$	Replaced-token detection	33 GB	82.4 [57]
$\text{DeBERTa}_{\text{base}}$	Disentangled MLM	78 GB	86.8 [59]

2.3.3 Fine-Tuning Strategies and Best Practices

Although a Transformer encoder already encodes extensive linguistic knowledge, naïvely updating all weights can lead to catastrophic forgetting or over-fitting on small data sets. Researchers have therefore devised optimisation heuristics that preserve pre-training benefits while adapting to new labels.

Layer-wise learning-rate decay. Lower layers capture generic lexical and syntactic information, whereas upper layers specialise for downstream tasks. Applying a multiplicative decay $\eta_\ell = \eta_0 \cdot \gamma^{L-\ell}$ (with $\gamma \approx 0.95$) keeps early layers close to their pre-trained state [60]. Empirically this yields +1-2 GLUE points over a flat learning rate.

Discriminative learning rates and gradual unfreezing. Howard and Ruder [43] first showed-in ULMFiT for LSTMs-that using different η per layer, combined with *gradual unfreezing* (start fine-tuning from the top layer and unfreeze one lower layer each epoch), stabilises training on data sets with as few as 1 000 examples. The same schedule transfers well to BERT and RoBERTa [60].

Regularisation techniques. *Mixout* stochastically replaces a fraction of weight vectors with their pre-trained values during back-propagation and performs on-par with dropout while preserving knowledge [61]. Other effective choices are weight decay (0.01) and label smoothing (0.1) [62].

Task-adaptive pre-training. If unlabelled in-domain text is available, an extra MLM pass on that corpus-often called continued or TAPT-bridges domain shift and can lift F1 by 3-5 points, especially for specialised jargon [60].

Typical hyper-parameters. Table 2.2 lists values that work well across sentiment, topic, NER, and entailment tasks; they serve as defaults in our experiments.

A small set of tuning rules-layer-wise decay, discriminative rates, robust regularisation, and, when possible, task-adaptive pre-training- is sufficient to turn a generic encoder into a

Table 2.2: Recommended fine-tuning settings for BERT-base-like models.

Parameter	Value
Batch size	32 (or 16 if memory-bound)
Max sequence length	128 (512 for long documents)
Base learning rate η_0	2×10^{-5}
LR scheduler	linear warm-up 10 % $\rightarrow$ decay to 0
Layer-wise LR decay γ	0.95
Weight decay	0.01
Dropout / mixout	0.1 / 0.9 keep rate
Epochs	3 (early-stop on dev loss)

state-of-the-art text classifier with only a few thousand labelled examples [60, 62]. We adopt these defaults in all downstream experiments that follow.

2.4 Contrastive Representation Learning

Cross-entropy fine-tuning (2.3.3) separates classes in probability space yet leaves the geometry of the embedding space undefined. Contrastive objectives remedy this by simultaneously optimising *alignment* of positive pairs and *uniformity* of the overall distribution, producing vectors where cosine distance tracks semantic similarity more faithfully [63]. Such geometry directly benefits the prototype and nearest-neighbour methods of 2.2.2.

2.4.1 Triplet loss

The classical formulation is the triplet hinge loss [64, 65]

$$\mathcal{L}_{\text{triplet}} = \max\bigl\{0,\ d(\mathbf{a},\mathbf{p}) - d(\mathbf{a},\mathbf{n}) + m\bigr\},$$

where an anchor $\mathbf{a}$ must be at least a margin $m > 0$ closer to a positive $\mathbf{p}$ than to a negative $\mathbf{n}$. Although conceptually simple, effective training depends on carefully sampling *hard negatives*; most random triplets are already satisfied and yield no gradient, while extremely hard negatives slow convergence or destabilise optimisation [66, 67].

2.4.2 InfoNCE / NT-Xent

Modern practice therefore adopts the normalised temperature-scaled cross-entropy (NT-Xent) loss [68, 69]:

$$\mathcal{L}_i = -\log \frac{\exp\bigl(\operatorname{sim}(\mathbf{z}_i,\mathbf{z}_{i^+})/\tau\bigr)}{\sum_{k=1}^{2B}\mathbf{1}_{[k\neq i]}\exp\bigl(\operatorname{sim}(\mathbf{z}_i,\mathbf{z}_k)/\tau\bigr)},$$

where each anchor $\mathbf{z}_i$ is paired with a single positive $\mathbf{z}_{i^+}$ and contrasted against all $2B-2$ in-batch negatives at temperature τ. Compared with the triplet loss, NT-Xent offers four advantages:

1. *Implicit hard negatives*: large batches provide a natural hardness spectrum without an explicit mining phase [68].
2. *Smooth gradients*: the softmax denominator gives every negative a non-zero contribution, avoiding the dead-zone problem of the hinge [69].
3. *No margin tuning*: only a single temperature hyper-parameter is required, while triplet performance is sensitive to the margin m [65].
4. *Theoretical guarantees*: NT-Xent provably optimises upper and lower bounds on alignment and uniformity [63].

A supervised extension, *Supervised Contrastive Learning*, treats all samples sharing the anchor's label as additional positives, further tightening class clusters [13].

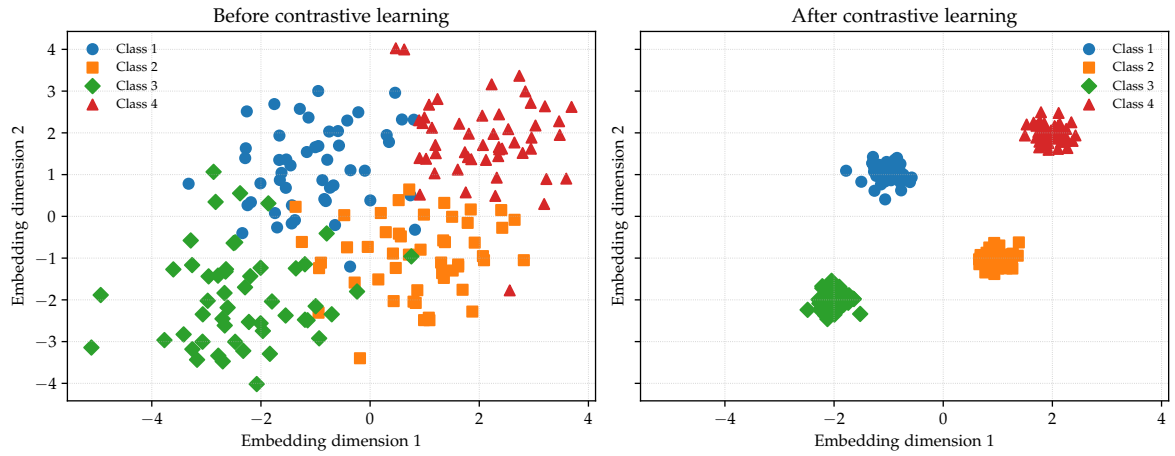

Figure 2.7: Conceptual effect of contrastive training on a two-dimensional embedding: positives collapse, negatives repel, enlarging margins.

The geometry in Figure 2.7 implies that one may retain multiple local prototypes instead of averaging them into a single centroid, yielding non-spherical decision regions in high dimension without training an additional classifier. Thus, contrastive-tuned embeddings provide stronger, margin-maximising inputs for downstream prototype and k-NN decision rules.

2.5 Weakly-Supervised Learning and Pseudo-Labeling

Contrastive learning reshapes an embedding space *without* additional human labels. Yet many NLP tasks still require *some* form of supervision to map vectors to semantic categories. When expert annotation is scarce, practitioners turn to *weak supervision*: programmatic heuristics, distant alignment with knowledge bases, or large language-model prompts that emit noisy-but inexpensive-labels [5]. A related strategy, *pseudo-labeling*, iteratively trains a

model on a small seed set, predicts high-confidence labels for the unlabeled pool, and adds these predictions back into the training data [12]. Despite their noise, such label surrogates regularise deep encoders and unlock performance that would otherwise require tens of thousands of gold annotations [70]. This section formalises these ideas, reviews canonical workflows-including label propagation [71] and the topic-guided *LOTClass* [72] that we later employ as baselines-and critiques their advantages and pitfalls in low-resource text classification.

2.5.1 Definitions and Sources of Weak Labels

Weak supervision generates noisy but automatically scalable labels in place of costly expert annotation. Common sources include heuristic patterns, external knowledge bases (*distant supervision*), and ensemble labeling functions [5, 73]. *Pseudo-labeling* trains an initial model on a small seed set, predicts labels for unlabeled instances, keeps only high-confidence predictions, and retrains on the expanded data [12]. This bootstrapping loop has been scaled to billions of examples in the Noisy Student paradigm [70].

Figure 2.8 illustrates the pipeline: raw documents flow through multiple weak labelers whose outputs are aggregated before training a classifier. Label propagation on a k-NN graph spreads seed labels to neighbouring embeddings [71], while LOTClass derives pseudo labels from class-specific keywords [72]. These techniques serve as baselines and initialization for our proposed framework.

2.5.2 Pseudo-Labeling Workflows and Advanced Variants

Let $\mathcal{D}_L = \{(x_i, y_i)\}$ be a small seed set of gold labels and $\mathcal{D}_U = \{u_j\}$ a large pool of unlabeled examples. *Pseudo-labeling* trains an initial model $\mathcal{M}^{(0)}$ on $\mathcal{D}_L$, predicts labels $\hat{y}_j^{(t)}$ for $\mathcal{D}_U$, selects high-confidence examples

$$\mathcal{S}^{(t)} = \big\{ (u_j, \hat{y}_j^{(t)}) \;\big|\; \max_k p^{(t)}(y{=}k \mid u_j) \geq \tau \big\},$$

adds $\mathcal{S}^{(t)}$ to the training set, and retrains to obtain $\mathcal{M}^{(t+1)}$ [12]. Iteration stops when no new samples exceed the confidence threshold τ or a maximum epoch count is reached. The loop is robust in practice: Xie, Luong, Hovy, and Le [70] scaled it to 300M images, while Arazo, Ortego, Albert, et al. [74] showed that temperature sharpening and Gaussian-mixture uncertainty estimates reduce confirmation bias.

Label propagation constructs a k-nearest-neighbour graph whose edge weights encode similarity. The harmonic solution

$$\ell^* = \arg\min_{\ell} \sum_{(i,j)\in E} w_{ij} \, (\ell_i - \ell_j)^2 \tag{2.5}$$

spreads seed labels to spectral clusters [71]. *LOTClass* maps class keywords into embedding space and uses anchor words with Gumbel-softmax self-training to produce competitive pseudo labels [72].

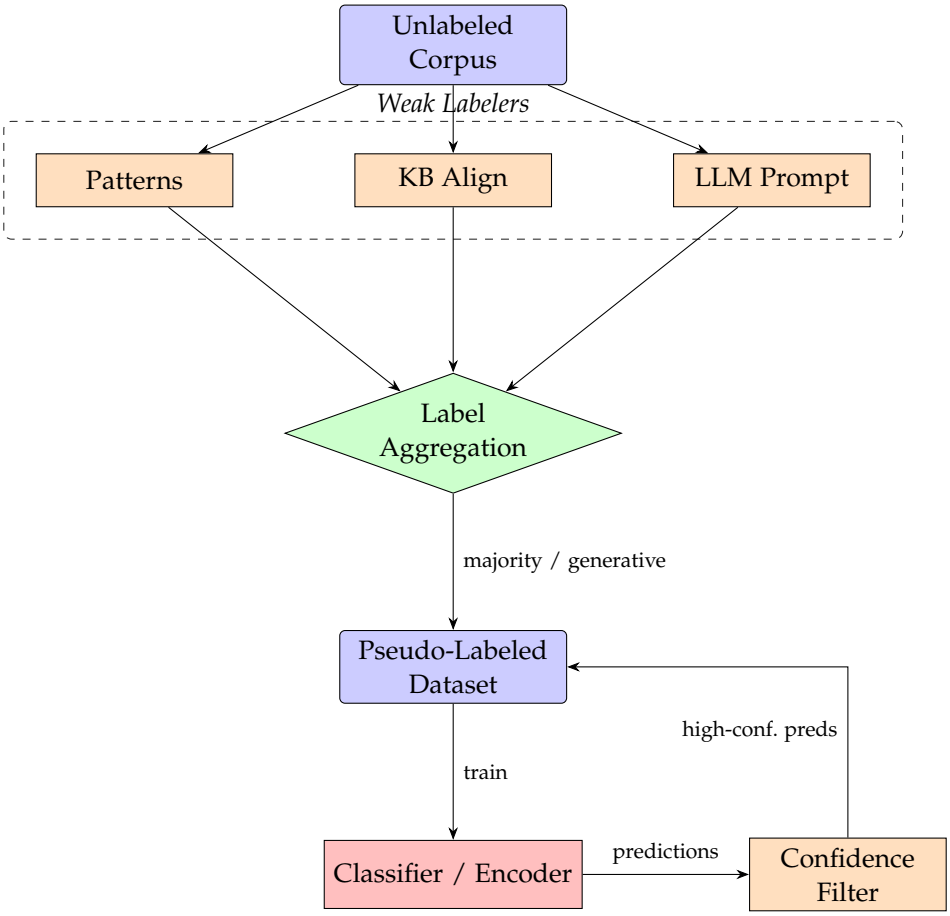

Figure 2.8: Weak supervision and pseudo-labeling workflow. Documents pass through an ensemble of weak labelers; aggregated labels form a pseudo-labeled set that trains a classifier whose high-confidence predictions iteratively enlarge the training data.

Take-aways. Weak supervision eliminates most annotation cost but introduces noise that can propagate confirmation bias and class imbalance [5, 74]. Confidence thresholds and temperature sharpening mitigate these risks. These methods—confidence-based self-training, label propagation, and LOTClass—serve as baselines and warm-start initializers for our label-efficient framework.

2.6 Zero- / Few-Shot Prompting with Large Language Models

Large language models (LLMs) are essentially deep Transformer decoder stacks, scaled to hundreds of billions or even trillions of parameters-orders of magnitude beyond encoder-only models such as BERT (110 M parameters) [40, 41]. GPT-3, with its 175 B parameters, first demonstrated that a model of such scale could perform diverse tasks in a zero- or few-shot manner using only natural-language prompts [10]. Its successor, GPT-4, is estimated

to harness around 1.8 T parameters, exhibiting human-level performance on professional benchmarks [75, 3]. The February 2025 preview of GPT-4.5 further advances capabilities on text and multimodal tasks [76], while GPT-4o ("omni"), launched in late 2024, integrates text, audio, vision, and video inputs and outputs in real time [77]. Anthropic's Claude-3 family, released in early 2024, also targets the trillion-parameter regime and achieves state-of-the-art conversational and reasoning performance [4]. At this scale, LLMs internalize vast world knowledge and implicit reasoning heuristics, enabling them to act as *noisy oracles*-solving classification, QA, and generation tasks without any gradient-based fine-tuning, solely through prompt design [78].

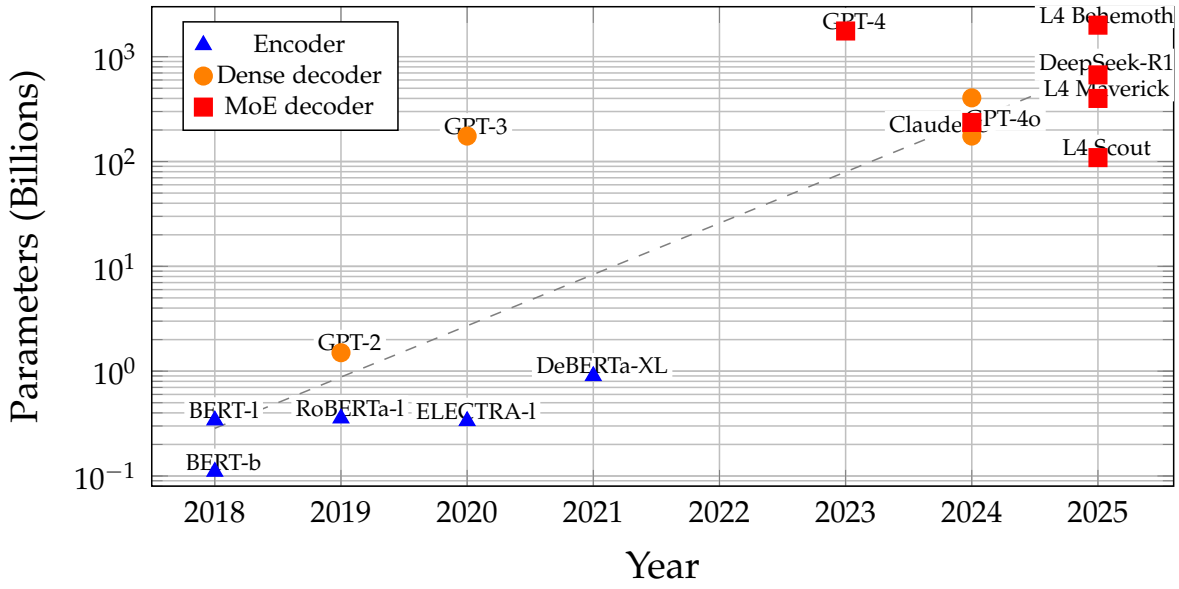

Figure 2.9: Trajectory of parameter scaling from early Transformer encoders to modern LLMs

2.6.1 Prompting Fundamentals

In large language models, a *prompt* is a natural-language instruction that specifies a task, often accompanied by contextual information or examples. At its simplest, a prompt might read:

```
"Classify the sentiment of the following review: '[REVIEW_TEXT]'."
```

More elaborate prompts introduce a *role*, delimiters, and explicit formatting:

```
"You are a helpful assistant. Identify the sentiment (Positive or Negative)
of the review below:
--
'[REVIEW_TEXT]'
Reply with one word only."
```

Designing effective prompts-often called *prompt engineering*-requires careful choice of instruction wording, delimiters, and (for few-shot setups) selection of exemplars. Small variations in phrasing or format can substantially alter model behaviour, as demonstrated in systematic surveys of prompt techniques [78, 10].

2.6.2 Zero-Shot Prompting

In the zero-shot setting, a task is specified solely through natural-language instructions, with no demonstration examples provided in the prompt. A typical zero-shot classification template might be:

```
"Determine whether the following product review expresses a Positive or Negative
sentiment:
--
'[REVIEW_TEXT]'
Reply with "Positive" or "Negative" only."
```

State-of-the-art LLMs are pretrained on massive web-scale corpora, enabling them to map such instructions to accurate predictions without gradient updates [10]. Instruction tuning further aligns models to follow natural directives, narrowing the zero-shot performance gap with fine-tuned systems [79]. For complex reasoning tasks, appending the phrase "`Let's think step-by-step`" elicits intermediate chain-of-thought outputs that improve accuracy on arithmetic and commonsense benchmarks [80]. To reduce variability, multiple reasoning paths can be sampled and aggregated via self-consistency, which ensembles answers across diverse chains [81].

Throughout this thesis, we employ zero-shot prompting both as a *baseline classification method* and as the means to distill *semantic archetypes*-concise, instruction-derived protoypes of each class-which serve as the starting point for our annotation pipeline in Chapter 4.

2.6.3 Few-Shot In-Context Learning

Few-shot in-context learning (ICL) augments the prompt with a small number of exemplar input-output pairs, enabling the LLM to induce the task mapping directly from the context. For classification, a typical prompt looks like:

```
"Classify the sentiment of these movie reviews.
Review:  'A heartwarming tale of friendship and hope.'  → Positive
Review:  'Predictable plot and wooden acting.'  → Negative
Review:  '[NEW_REVIEW]' →"
```

Providing k such examples (often $k = 4$-8) yields accuracy gains of 10-30 points over zero-shot on GLUE-style benchmarks [82]. LM-BFF further improves few-shot performance by automatically selecting and reformulating exemplars and tuning label verbalizers [82].

Despite these strengths, ICL remains sensitive to prompt variation: reordering examples or changing punctuation can shift accuracy by over 15 points [83]. Token budget constraints

on hosted APIs force careful exemplar selection or compression, especially for long contexts. Hallucination and bias also persist: the model may produce fluent but incorrect labels or propagate stereotypes encoded in the exemplars [84, 74]. Mitigations include uncertainty-based filtering of low-confidence predictions [74] and self-consistency decoding that aggregates outputs over multiple exemplar orders [81].

In this thesis, few-shot ICL is used solely as a *baseline classifier*. We supply a few examples per class in the prompt and compare its performance to our weakly-supervised and contrastively-tuned methods. While ICL can approach fully supervised accuracy with minimal labeling effort, its instability and cost per query underscore the need for more robust, label-efficient pipelines.

2.7 Reinforcement-Learning Feedback for NLP Models

Classical supervised fine-tuning adjusts a language model to match static labels, but it cannot incorporate *interactive* preferences that emerge after deployment. Reinforcement learning (RL) provides a complementary framework: an *agent* (the model) selects actions a_t in a state s_t and receives delayed rewards r_t, iteratively improving its policy to maximise expected return [85]. Policy-gradient methods-starting with REINFORCE [86]-directly optimise the model's parameters by ascending the gradient of this return. Modern variants such as Generalised Advantage Estimation (GAE) [87] and Proximal Policy Optimisation (PPO) [14] stabilise training and have become the engine of *reinforcement learning from human feedback* (RLHF) [79]. RLHF fine-tunes GPT-4-class models to better align with user intent, while recent work extends the paradigm with AI-generated feedback (RLAIF) and group-relative policy optimisation (GRPO) to reduce human annotation cost; DeepSeek-R1 demonstrates that such automated RL can yield state-of-the-art reasoning ability at a fraction of the supervised data budget [23]. Later we adopt the same principle at a smaller scale: classifier-derived rewards are fed back to an LLM to refine our semantic archetypes distillation, closing the loop between static prompts and dynamic performance.

2.7.1 Reinforcement-Learning Fundamentals

Reinforcement learning formalises the interaction between an *agent* and its *environment* as a Markov decision process (MDP). At each time-step t, the agent observes a state $s_t \in \mathcal{S}$, samples an action $a_t \sim \pi_\theta(\cdot \mid s_t)$ from its stochastic policy π_θ with parameters θ, receives a scalar reward $r_t \in \mathbb{R}$, and transitions to the next state s_{t+1}. The objective is to maximise the expected return,

$$J(\theta) \;=\; \mathbb{E}_{\pi_\theta}\Big[\, G_t \Big], \qquad G_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k},$$

where $\gamma \in [0, 1)$ is a discount factor [85].

Policy-gradient theorem. The seminal REINFORCE algorithm derives an unbiased gradient estimator for $J(\theta)$:

$$\nabla_\theta J(\theta) \;=\; \mathbb{E}_{\pi_\theta}\big[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t\big],$$

allowing the agent to update its parameters via stochastic gradient ascent [86]. In practice, subtracting a *baseline* $b(s_t)$ that does not depend on the action reduces variance without introducing bias, paving the way for actor-critic architectures.

Why RL matters for NLP. Language models can be cast as policies that generate token actions conditioned on a textual state (the context). When rewards capture human preferences-fluency, factuality, or task success-policy gradients permit direct optimisation of these objectives instead of maximising likelihood alone. The next subsection introduces *generalised advantage estimation* and *proximal policy optimisation*, two algorithms that address the high variance and instability of vanilla REINFORCE and have become standard for aligning large language models with human or automated feedback.

2.7.2 Efficient Policy Optimisation: GAE and PPO

Vanilla REINFORCE (Equation (2.7.1)) suffers from high variance because every sampled return G_t is used wholesale as the gradient weight. Actor-critic methods replace G_t with an *advantage* estimate $A_t = G_t - V_\phi(s_t)$ that subtracts a learned baseline V_ϕ, but the variance-bias trade-off remains. Schulman, Moritz, Levine, et al. [87] address this with *generalised advantage estimation* (GAE), a weighted exponentially decaying sum of k-step temporal-difference errors:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \big(r_{t+l} + \gamma V_\phi(s_{t+l+1}) - V_\phi(s_{t+l})\big),$$

where $\lambda \in [0,1]$ controls the bias-variance continuum. Setting $\lambda = 0$ recovers high-bias, low-variance TD(0); $\lambda \to 1$ approaches REINFORCE.

Building on GAE, Schulman, Wolski, Dhariwal, et al. [14] introduced *proximal policy optimisation* (PPO), which limits each policy update to a small, trustworthy region by clipping the probability ratio $r_t(\theta) = \pi_\theta(a_t|s_t) / \pi_{\theta_{\text{old}}}(a_t|s_t)$:

$$\mathcal{L}^{\mathrm{PPO}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\,\hat{A}_t,\; \mathrm{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\,\hat{A}_t\big)\Big] - \beta\,\mathrm{KL}\big[\pi_{\theta_{\text{old}}} \| \pi_\theta\big],$$

where $\epsilon \approx 0.1-0.2$ and the KL penalty β guards against overly large steps. PPO achieves state-of-the-art sample efficiency while remaining simple to implement, and has become the de-facto optimisation method for *reinforcement learning from human feedback* (RLHF) [79]. Large language models such as GPT-4 and Claude-3 are aligned via a reward model trained on human preference pairs, then fine-tuned with PPO using KL divergence to the supervised policy as a secondary safety constraint. Recent systems like DeepSeek-R1 further adapt PPO with group-relative objectives to enhance reasoning quality [23].

In our pipeline, the same mechanism will drive a lightweight feedback loop: rewards derived from classifier performance are fed back to the archetype-generating language model, allowing it to refine class representations while staying within a controlled KL distance of the initial prompt-based policy.

2.7.3 RL with Human and Automated Feedback

Large language models generate sequences token-by-token, which makes them amenable to *policy* optimisation with the algorithms above. The current standard pipeline-often dubbed *reinforcement learning from human feedback* (RLHF)-proceeds in three stages [79]:

1. **Supervised fine-tuning**: start from a pretrained model and fit it to a small set of instruction-response pairs, yielding a policy π_{θ_0}.
2. **Reward modelling**: collect preference pairs $(y_{\text{good}}, y_{\text{bad}})$ ranked by humans (or another evaluator) and train a reward function R_ψ such that $R_\psi(y_{\text{good}}) > R_\psi(y_{\text{bad}})$.
3. **Policy optimisation**: use PPO with a KL-penalty to the reference policy to maximise the expected reward $\mathbb{E}_{y\sim\pi_\theta}\big[R_\psi(y)\big]$, producing an aligned $\pi_{\theta^\star}$.

RLHF elevated GPT-3.5 and GPT-4 to state-of-the-art instruction following, but human preference collection is costly. Two extensions reduce this burden. *RLAIF* (Reinforcement Learning from AI Feedback) replaces human annotators with a committee of trusted models that critique and rank candidate outputs [88]. *Group Relative Policy Optimisation* (GRPO) goes further: it trains several policies in parallel and rewards each for outperforming the group median, encouraging competition and diversity [23]. DeepSeek-R1 employs GRPO with 5 B-parameter students and shows that purely automated feedback can improve reasoning benchmarks by 7-10 F1 while maintaining factual consistency.

3 Background and Related Work

Chapter 2 developed the theoretical toolkit for these thesis: vector spaces, prototype classifiers, prompt-based supervision, and reinforcement-learning feedback. We now turn from theory to the concrete project context: *CreateData4AI* (CD4AI), a research project at TU Munich that aims to transform raw, unlabeled text into structured and annotated datasets with minimal human effort.[1] Guided only by a domain expert's class list and a handful of seed keywords, CD4AI automatically expands the vocabulary for each class, extracts and parses context windows around those terms, clusters similar usage patterns, and distils every cluster into a concise *semantic archetype*. The archetypes act as interpretable prototypes that downstream components use for corpus annotation.

This chapter has three roles. **(i)** Section 3.1 outlines the complete CD4AI pipeline, indicating where automation ends and expert input begins. **(ii)** Section 3.1 examines each preprocessing component-keyword expansion, context extraction, pattern clustering, and archetype synthesis-detailing the hyperparameters adopted in this thesis and highlighting the reinforcement-learning refinement that constitutes our main contribution. **(iii)** Section 3.2 situates CD4AI within prior research on prototype-driven classification, weak supervision, and LLM-based data creation, providing the further support for the methodology and experiments that follow.

3.1 The CreateData4AI Pipeline

CreateData4AI (CD4AI) turns an unannotated corpus into training data through a four-stage, bottom-up workflow that begins with a domain expert's class list and a handful of *seed keywords*. Each stage raises the abstraction level while preserving an option for expert validation:

1. **Iterative Keyword Expansion**. Dense-embedding similarity grows each seed list to ≈ 50 domain-specific terms per class, over three iterations with a similarity threshold of 0.7 [89].

2. **Context Window Extraction**. For every expanded keyword occurrence, a dependency parser selects a window of three to ten tokens by traversing at most three dependency hops, yielding linguistically coherent *context windows*.

3. **Recursive Hierarchical Clustering**. Context windows are embedded and clustered with Ward linkage; an adaptive distance threshold and density check produce 10-30 context window clusters per class.

[1] `https://wwwmatthes.in.tum.de/pages/nqpi6qljq0x9/CreateData4AI-CD4AI`

4. **LLM-based Archetype Distillation**. An instruction-prompted Llama-3 8B model converts each cluster into a concise *semantic archetype* that includes a natural-language description and two or three concrete examples.

A domain expert may inspect or edit the outputs of any stage-approving newly proposed keywords, discarding irrelevant context windows, merging or splitting clusters, or rewriting archetype phrasings-but the thesis assumes the fully automatic settings above to ensure comparability of experimental results. The distilled archetypes form interpretable prototypes that the next chapter's methodology will leverage for large-scale annotation and classifier training.

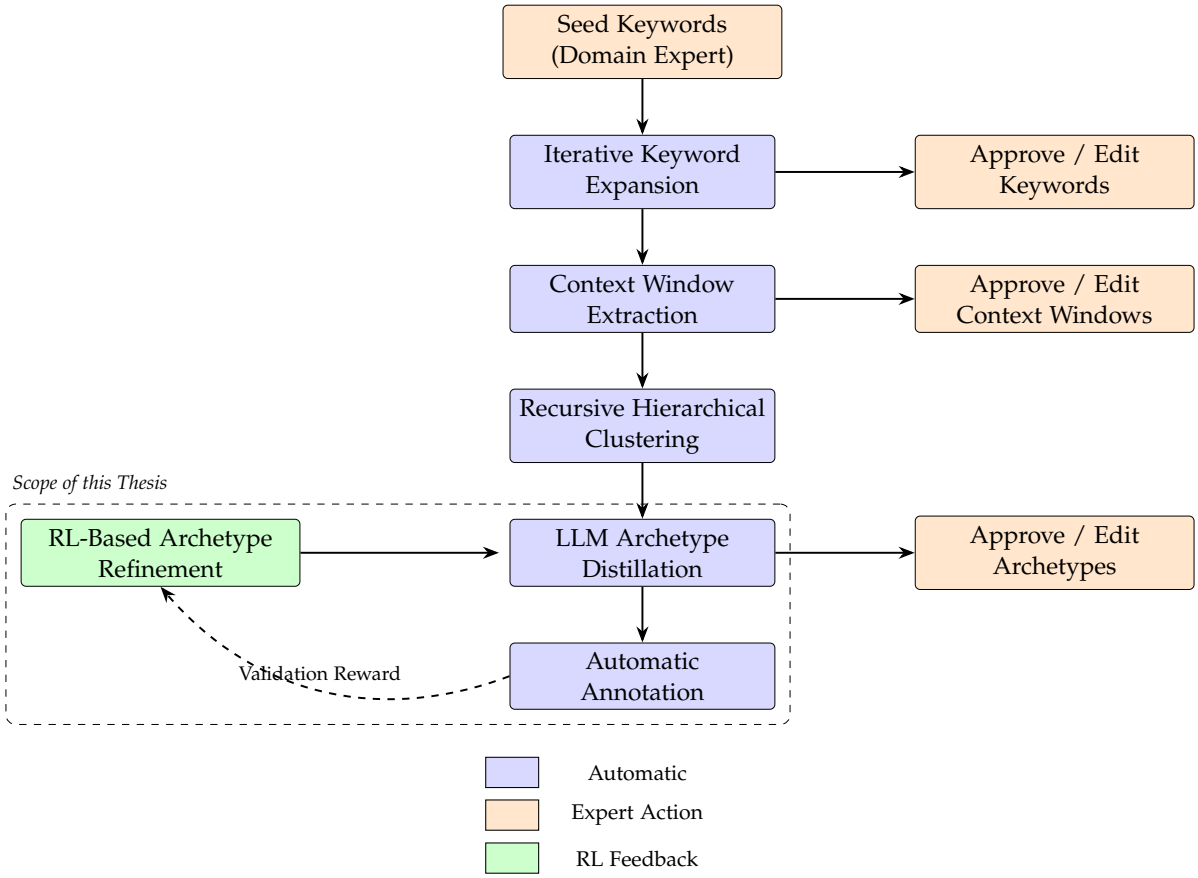

Figure 3.1: Processing Pipeline in CreateData4AI

3.1.1 Project Scope and Inputs

CD4AI addresses domains where large quantities of unlabelled text exist but expert-labelled data are scarce. The pipeline begins with two artefacts provided by a domain specialist:

- **Class list**. A fixed set of target categories such as *Politics*, *Economy*, or *Sports*.

- **Seed keywords**. Three to five representative terms for each class-for *Politics*, for example, *election*, *campaign*, *ballot*.

From these inputs the system expands each seed list, extracts context windows around every expanded term, clusters similar windows, and finally distils each cluster into a concise *semantic archetype*. A dashboard lets the expert approve or refine intermediate results at every stage, but for the experiments reported in this thesis we evaluate the fully automated path to ensure reproducibility and isolate the impact of the proposed reinforcement-learning refinement introduced later. The next subsections detail each component and the Hyperparameters adopted in this study.

3.1.2 Iterative Keyword Expansion

Purpose The three-to-five seed terms provided for each class are precise yet incomplete. Expanding them ensures that later stages capture the full lexical variety of the domain while remaining semantically focused.

Algorithm The corpus is split into five equal chunks. On each chunk the procedure applies two complementary similarity measures[2] : `extract_keywords_max` picks candidates maximally similar to any seed keyword, whereas `extract_keywords` compares each candidate to the mean embedding of the seed set. Both rely on cosine similarity in the `jinaai/jina-embeddings-v3` space. After scoring, the 99th percentile threshold retains the strongest candidates; up to five new terms are appended to the seed list before the next iteration. After five iterations each class accumulates roughly 30-50 keywords.

Parameter	Value / Setting
Embedding model	`jinaai/jina-embeddings-v3`
Iterations, n_{iter}	5
Percentile threshold	99
Max new seeds per iteration	5
Seed : document weight	1 : 0 (seed similarity only)
LOF / ISO contamination	0.5
ConceptNet similarity cut-off	0.80

Table 3.1: Key Hyperparameters for the keyword-expansion component.

Advanced filtering All extracted terms are embedded and clustered with Ward linkage. Clusters lacking any original seed word are removed via a convex-hull test, and residual outliers are pruned with Local Outlier Factor and Isolation Forest (contamination 0.5). A final ConceptNet check eliminates terms whose similarity to the seed-plus-synonym set falls below 0.80.

[2]Implementation repository: `https://github.com/sjmeis-cske/keyword_extractor`.

Output The component returns a mapping $\mathcal{K} = \{c \mapsto K_c\}$ from each class c to its expanded vocabulary. Tuning the expander beyond these published settings lies outside the scope of this thesis; any alternative choices would alter absolute scores but not the relative comparison of annotation and classification methods in later chapters.

3.1.3 Context-Window Extraction

Purpose After vocabulary expansion, the pipeline must harvest local evidence showing how each keyword functions in running text. These *context windows* must be short enough for efficient clustering yet syntactically complete enough to preserve meaning.

Sentence processing Documents are lower-cased and split into sentences with the NLTK tokeniser. Sentences containing at least one class keyword are parsed with the `en_core_web_md` spaCy model; switching to another language merely requires the corresponding spaCy pipeline, so the component remains language-agnostic.

Anchors, links, and hops In a dependency tree each arc connects a *dependent* token to its *head*. Following a head link moves *up* the tree, while following dependents moves *down*. We call one such step a *hop*. Limiting traversal to three hops in either direction captures the immediate syntactic neighbourhood without drifting into distant clauses, a practice widely used in dependency-based extraction [90].

Multi-token keywords (our addition) Earlier versions of the CD4AI pipeline handled only single-token anchors. We extend the algorithm to accept multi-token phrases (e.g. *climate policy*). The first token (*climate*) serves as the anchor; dependency expansion almost always includes the remaining tokens, so the full phrase appears in the final window.

Window growth Starting from the anchor token the algorithm

1. follows head links up to three hops, expanding window boundaries,
2. then follows dependent links up to three hops.

The provisional span is re-centred so the anchor is at most ten tokens from either edge; windows shorter than three words are discarded.

> *Sentence*: "After months of debate, the **climate** policy was finally approved by parliament."
> *Anchor*: `climate`
> Upward hops: `climate` → `policy` → `approved`
> Downward hops: none
> Final window: *"the climate policy was finally approved"*

Filtering and limits After extraction, HTML tags, punctuation, and extra spaces are removed; duplicate windows are eliminated. Extraction stops once 50 000 windows have been collected for a class. Batching (2 000 sentences) and multi-processing (CPU cores - 2) sustain throughput on million-document corpora.

Parameter	Value / Setting
spaCy model	`en_core_web_md` (swappable)
Max upward hops	3
Max downward hops	3
Max window radius	10 tokens
Min window length	3 words
Batch size	auto (CPU cores - 2)
Chunk size	2 000 sentences
Max windows per class	50 000

Table 3.2: Key Hyperparameters for context-window extraction.

Output For each class the extractor yields a set of syntactically coherent, keyword-centred windows that feed the clustering stage. Different parsers or hop limits would alter absolute counts, but every method evaluated later operates on the same extracted sample, so comparative results remain valid.

3.1.4 Recursive Hierarchical Clustering

Purpose The thousands of context windows extracted per class must be grouped into coherent usage patterns before an LLM can distil them into archetypes. Simple "flat" clustering either over-merges dense zones or fragments sparse ones. We therefore adopt a *recursive density-based hierarchical* approach that refines clusters until each region of embedding space is both compact and semantically homogeneous.

Algorithm outline

1. **Embedding**. All windows are encoded with `jinaai/jina-embeddings-v3`. Vectors are *not* L2-normalised, because Ward linkage assumes Euclidean distances on raw coordinates.

2. **Top-level tree**. Ward's minimum-variance linkage [21] builds a dendrogram in $O(n^2 \log n)$ time. The tree is cut at an initial threshold $\tau_0 = \textit{distance\_threshold\_factor} \times n$, producing broad clusters.

3. **Density test**. For each cluster the algorithm counts pairwise Euclidean distances below ε and computes

$$\text{density} \;=\; \frac{\#\{\, d < \varepsilon \,\}}{\binom{|C|}{2}}.$$

If the density $< \theta$ ($\theta = 0.1$) the cluster is deemed *sparse*.

4. **Recursive split**. Sparse clusters are reclustered with a threshold $\tau \leftarrow \tau/1.1$; recursion stops when density $\geq \theta$ or when the cluster is a singleton.

5. **Representative window**. For every final cluster the arithmetic centroid is computed and the member with the smallest cosine distance to that centroid is chosen as the *centre text*; using a real window keeps examples human-interpretable.

Example Suppose there exists a dataset with a hypothetical *Climate* class which yields 12 000 windows. With $\tau_0 = 1.0 \times 12\,000$ the first cut produces eight clusters. Two have density < 0.1; each is reclustered with $\tau = \tau_0/1.1$, giving three denser sub-clusters apiece. The recursion terminates at depth 3, resulting in 14 clusters whose sizes range from 8 to 73 windows. The centre window for one cluster could be *"reduce carbon emissions in manufacturing"*, which summarises the cluster's focus on industrial decarbonisation.

Parameter	Value / Setting
Embedding model	`jinaai/jina-embeddings-v3`
Linkage method	Ward (variance minimisation)
Distance threshold factor τ_0	1.0
Density radius ε	0.5 (Euclidean)
Density threshold θ	0.1
Recursive division factor	1.1
Representative metric	cosine distance to centroid
Max recursion depth	unconstrained (empirically ≤ 5)

Table 3.3: Key Hyperparameters for recursive hierarchical clustering.

Output The module returns a list of clusters, each stored with its centre text, size, density, recursion level, and member indices. Typical clusters contain 5 - 50 windows, providing a balanced granularity for archetype synthesis in the following stage. Because the method relies only on the embedding model and Euclidean distances, it remains language-agnostic: substituting a multilingual encoder seamlessly transfers the procedure to non-English corpora.

Computational Note Clustering 20 000 windows requires roughly two seconds per thousand windows on an 8-core CPU (distance matrix + linkage), using about 1 GiB RAM per 10 000 windows. Recursion increases cost modestly, because only sparse sub-clusters are reclustered.

3.1.5 LLM-Based Archetype Distillation

What is a Semantic Archetype? Given a cluster of context windows that all instantiate the same latent theme, an *archetype* is a compact textual description-typically one concise sentence-plus two or three illustrative examples that capture the cluster's semantic core.

Archetypes compress hundreds of windows into a few human-readable "prototypes", enabling downstream classifiers to reason over a handful of abstractions instead of raw text.

Why an LLM? Large language models internalise vast linguistic knowledge and can generalise beyond the literal wording of the input. Prompting an LLM to "explain" a cluster therefore yields descriptions that are both faithful to the evidence and stylistically coherent, a capability demonstrated for tasks such as concept induction and rule extraction [10].

Prompt design and batching For each class the distiller feeds the model a system prompt that

- casts the LLM as a *domain expert* asked to write one archetype per cluster,
- instructs it to output strict JSON with keys `cluster_number`, `rule`, and `examples`,
- forbids meta-phrases such as "*this cluster describes*".

Clusters are appended as *"Cluster k: [window$_1$, ..., window$_{20}$]"*. Because Llama-3 8B has a 8 k-token context window, clusters are grouped so that each batch plus prompt fits within a configurable `token_limit` (default 1500). If a single cluster exceeds the limit, only its first 20 windows are sent.

LLM Generation Generation uses the HuggingFace `text-generation` pipeline with **deterministic** decoding (`do_sample = False`) and a repetition penalty of 1.2. The maximum number of new tokens is set dynamically:

$$\text{max\_new} = \lfloor \text{len(input)} \times \text{max\_new\_tokens\_factor} \rfloor.$$

With the default factor 1.0, output length scales linearly with input size but never exceeds the model limit.

Robust JSON Extraction LLM outputs often contain extra quotation marks, missing commas, or trailing commas. A post-processor locates every "{ ... }" span, applies regular-expression fixes, and validates that each object has the required three keys. Malformed clusters are logged and marked as "missing" in the statistics.

Representative Metadata For every cluster the distiller stores:

- `archetype_text` - the generated rule
- `examples` - up to three model-generated sentences
- `source_texts` - all original windows in the cluster
- `cluster_size`, `density`, recursion level, and batch id.

Parameter	Value / Setting
LLM model	`meta-llama/Meta-Llama-3-8B-Instruct`
Token limit per batch	1500
max_new_tokens_factor	1.0
Repetition penalty	1.2
Sampling	disabled (`do_sample = False`)
JSON parser tolerance	fixes quotes, commas, comments

Table 3.4: Key Hyperparameters for archetype distillation

Example Output

{"cluster_number": 7, "rule": "*Factories adopting clean energy technologies*", "examples": ["Solar panels now power the assembly line.", "The plant switched from coal to biomass fuel."]}

Language Model Agnosticism Although Llama-3 is used here, any instruction-tuned LLM can replace it. The only requirements are JSON compliance and enough context length to handle batched clusters, making the component compatible with multilingual or domain-specific models.

Output The distiller returns an object containing all archetypes by class, overall statistics, configuration, and processing time. No hard cap is imposed on the number of archetypes; every cluster receives a prototype, ensuring complete coverage for the classifier stage.

The preceding subsections have surveyed every stage of the existing CD4AI preprocessing pipeline-keyword expansion, context-window extraction, recursive clustering, and LLM-based archetype generation. What remains is to label the corpus by assigning documents to the most appropriate archetypes, a task that reduces to text classification and is addressed in Chapter 4. Before turning to that methodology, however, the next section positions our approach within the broader landscape of related work.

3.2 Related Work

This section positions CD4AI within three research lines: prototype-centric classification and metric learning (3.2.1), weakly supervised labeling pipelines (3.2.2), and LLM-driven data creation together with policy-feedback refinement (3.2.3). Across all three, we emphasise methods that either reduce annotation effort or produce interpretable decision criteria, aligning with the project goal of turning raw corpora into class-wise semantic archetypes and labels with minimal manual supervision.

3.2.1 Prototype-Centric Classification and Metric Learning

Classical IR framed category decisions as proximity to class representatives: Rocchio constructs a prototype by pushing the class centroid toward relevant documents and away

from non-relevant ones, while the nearest-centroid (nearest class mean) classifier assigns a document to the closest class mean under a chosen metric. These methods motivate our use of distance geometry and class representatives in vector space. [55]

Modern metric-learning generalises this idea. Prototypical Networks learn a metric space in which each class is represented by the mean of its support embeddings; Matching/Induction Networks and memory-based variants build stronger class-wise representations for few-shot NLP. Recent analyses show that prototype-based networks can improve robustness and interpretability for text classification. These results support our later choice to compare documents to *semantic archetypes* via cosine distance rather than train a heavy parametric classifier. [91]

3.2.2 Weak Supervision for Text Classification

Weak supervision replaces expensive gold labels with programmatic or distantly-aligned signals, then aggregates them into probabilistic labels for model training. Data Programming (Snorkel) formalised this approach with labeling functions and generative label models; LOTClass showed that class keywords plus a masked-LM can bootstrap topic labels without annotations; label propagation spreads seed labels over a similarity graph; distant supervision aligns text with KB facts. Our pipeline likewise begins from seeds and derives weak signals, but we differ in: (i) producing interpretable, cluster-grounded *archetypes* before annotation, and (ii) later refining them with feedback from downstream performance. [5, 72, 71, 92]

3.2.3 LLM-Driven Data Creation and Policy-Feedback Refinement

Large language models enable zero/few-shot labeling via carefully designed prompts and instruction tuning. Self-Instruct shows that models can synthesize task exemplars that then supervise themselves; instruction-tuned models (e.g., InstructGPT) learn to follow natural prompts and provide high-quality labels without gradient updates on the downstream task. [10, 93, 79]

Quality can be further improved by preference-based reinforcement learning: RLHF optimises policies from human preferences (often with PPO). Recently, GRPO has emerged as an efficient online RL variant for verifiable/binary rewards and powered DeepSeek-R1 reasoning models, with theory connecting GRPO to a KL-regularised contrastive loss. These developments motivate our plan to feed a reward signal-derived from downstream classifier quality-back into the *archetype distillation* stage to iteratively improve our semantic archetypes. [14, 23]

4 Methodology

This chapter details the empirical framework used to validate the final *classifier* component of the CD4AI pipeline. Our goal is to show that the proposed classifier, together with its contrastive and weak-supervision refinements, achieves strong predictive performance and favourable efficiency on several benchmark corpora, while remaining label-efficient. To that end, we first outline the experimental setup (datasets, label splits, metrics, and compute environment), introduce the baseline systems against which we compare, and then provide methodological details for each of the four core approaches explored in this study.

4.1 Experimental Setup

This section specifies the conditions under which all experiments are run. We begin by describing the datasets and their train/dev/test partitions, followed by the label regimes (full supervision vs. 25-label-per-class low-resource setting), evaluation metrics, and hardware configuration. Subsequent sections (4.2-4.3) elaborate the baseline models and our proposed methods.

4.1.1 Datasets

All experiments use benchmark **English** corpora loaded from the `datasets` library on Hugging Face.[1] Focusing on English allows us to reuse a single set of language models during archetype distillation and classification; extending the CD4AI pipeline to other languages would merely require swapping tokenizer and embedding components. To keep GPU hours within a realistic budget we down-sample the training partitions and limit test sets to 1,000 samples. This sample size provides sufficient statistical power for reliable performance estimates while enabling extensive experimentation across multiple methods and datasets. No heavy cleaning is applied: after Unicode normalisation we tokenise with the default Byte Pair Encoding (BPE) or WordPiece tokenizer of the chosen encoder.

20 Newsgroups (`SetFit/newsgroup_20`) Usenet posts from 20 topical forums [16]. Long documents and class imbalance make it a classic stress-test; it is still used by BERT baselines and prototype studies alike.

[1] The exact dataset identifiers are given in Table 4.1. Relying on a common distribution channel simplifies reproducibility and version pinning.

Table 4.1: Corpus statistics and down-sampled splits used in this thesis. "Orig." denotes the canonical split sizes provided by the Hugging Face versions; all numbers are document counts. Test sets are limited to 1,000 samples for computational efficiency.

Dataset	Classes	Orig.		Our splits		
		Train	Test	Train	Dev	Test
20 Newsgroups	20	11 314	7 532	4 000	1 000	1 000
AG News	4	120 000	7 600	12 000	3 000	1 000
BBC News	5	1 225	1 000	1 225	300	700
DBpedia-14	14	560 000	70 000	14 000	3 500	1 000
arXiv	10	100 000	–	15 000	3 000	1 000

AG News (`ag_news`) Four-class news headlines dataset introduced by [17]. Short texts provide a complementary distribution to 20 NG.

BBC News (`SetFit/bbc-news`) RSS stories from five desks released for non-commercial research [18]. Given the dataset's limited size (1,225 training samples), we use the full training set without downsampling and partition the original 1,000-sample test set into 300 dev and 700 test samples to ensure proper hyperparameter tuning while maximizing training data utilization.

DBpedia-14 (`dbpedia_14`) Wikipedia abstracts mapped to 14 ontology classes [19]; a staple in large-scale classification benchmarks.

arXiv Categories (`effectiveML/ArXiv-10`) A curated subset of arXiv abstracts covering 10 major scientific categories: astrophysics (astro-ph), condensed matter (cond-mat), computer science (cs), electrical engineering and systems science (eess), high-energy physics phenomenology (hep-ph), high-energy physics theory (hep-th), mathematics (math), physics (physics), quantum physics (quant-ph), and statistics (stat). We use 15,000 abstracts for training with balanced sampling across categories, and custom dev/test splits of 3,000/1,000 samples respectively. All methods use these 10 main category labels for consistent evaluation.

Together, the five datasets cover forums, short news, encyclopædic summaries, and scientific abstracts, providing a diverse test-bed for the final classifier stage of CD4AI. Development sets are created through stratified random sampling (seed 42) from the original training data where not provided by the dataset, ensuring class balance for reliable hyperparameter selection.

4.1.2 Label Regimes

In line with the project's design, we adopt a *gold-label* regime of **25 documents per class** as the minimal supervision budget. This choice situates our evaluation squarely within the widely

used few/low-label settings in recent text-classification research: semi-supervised methods commonly study regimes between *10* and *50* labels per class (e.g., MixText reports results at 10, 200, and 2 500 labels/cls across standard benchmarks [94], while UDA demonstrates strong performance with as few as 20 labeled examples on text classification [95]). The **full train split** from Table 4.1 is always available as unlabelled data for pipeline stages (keyword expansion, clustering, archetype distillation) and for our classifier refinements.

4.1.3 Evaluation Metrics

We follow standard practice in text classification and report *Accuracy* and *Macro-F1* as our primary metrics, along with macro-averaged *Precision* and *Recall* for detailed analysis [51, 30, 56]. Let $\mathcal{C} = \{1, \ldots, K\}$ be the class set and let (x_i, y_i) be the i-th test instance with gold label $y_i \in \mathcal{C}$ and prediction $\hat{y}_i$. For each class $c \in \mathcal{C}$ we use the one-vs-rest view to define true/false counts: TP_c, FP_c, FN_c, TN_c.

Accuracy Overall fraction of correct predictions:

$$\mathrm{Acc} \;=\; \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[\hat{y}_i = y_i] \;=\; \frac{\sum_{c\in\mathcal{C}} \mathrm{TP}_c}{N}\,.$$

Per-class Precision, Recall, and F1 For class c:

$$\mathrm{Prec}_c \;=\; \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c}, \qquad \mathrm{Rec}_c \;=\; \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c}, \qquad \mathrm{F1}_c \;=\; \frac{2\,\mathrm{Prec}_c\,\mathrm{Rec}_c}{\mathrm{Prec}_c + \mathrm{Rec}_c}\,.$$

Macro-averaged Precision and Recall We report macro-averaged versions for overall performance assessment:

$$\mathrm{MacroPrec} \;=\; \frac{1}{K}\sum_{c\in\mathcal{C}} \mathrm{Prec}_c, \qquad \mathrm{MacroRec} \;=\; \frac{1}{K}\sum_{c\in\mathcal{C}} \mathrm{Rec}_c\,.$$

Macro-F1 (primary) Unweighted average over classes, which treats minority and majority labels equally and is therefore robust to imbalance:

$$\mathrm{MacroF1} \;=\; \frac{1}{K}\sum_{c\in\mathcal{C}} \mathrm{F1}_c\,.$$

Note on Micro-F1 In multiclass classification with single-label predictions (as in our setting), Micro-F1 equals accuracy by definition, since:

$$\mathrm{Prec}_\mu = \mathrm{Rec}_\mu = \mathrm{Acc} = \frac{\sum_c \mathrm{TP}_c}{N}\,.$$

Therefore, we omit Micro-F1 from our results tables to avoid redundancy with accuracy.

In addition to these predictive metrics, we will qualitatively comment on *efficiency* in terms of throughput (samples/s) and end-to-end latency measured on our stated hardware, keeping this discussion lightweight and comparable across methods.

4.1.4 Hardware and Runtime Environment

All GPU-accelerated training and local inference are executed on a Runpod virtual machine; LLM prompting baselines that rely on hosted models are executed via vendor Application Programming Interfaces (APIs) (e.g., OpenAI or comparable providers), with wall-clock latencies measured client-side. Unless stated otherwise, stochastic runs use a fixed random seed of 42. We use standard Hugging Face abstractions for dataset loading and model execution, and employ mixed-precision where supported. Throughput and latency figures reported in Chapter 5 are measured on the hardware below (local models) or as end-to-end request timings (API baselines).

Table 4.2: Compute environment for local training and inference.

Component	**Specification**
GPU	1 × NVIDIA RTX 5000 Ada
CPU	14 vCPU
System memory	62 GB RAM
Platform	Runpod VM instance
Precision	Mixed precision (automatic, where available)
Random seed	42 (training, data splits, and sampling)

4.2 Baselines

We compare the proposed classifier against prompting baselines that require no task-specific training and, where applicable, against supervised encoder baselines defined later in this chapter. This subsection specifies our zero- and few-shot LLM setups, model groups, and prompt formats.

4.2.1 LLM Prompting (Zero- and Few-Shot)

We evaluate proprietary and open-weight LLMs across four capacity tiers to probe accuracy-cost trade-offs. Each model is run in both *zero-shot* and *few-shot* settings with identical task phrasing; the few-shot variant draws in-context demonstrations from the Gold-25/cls pool (seed 42). Outputs undergo light post-processing (whitespace/quote trimming, case-folding, strict label matching) to prevent formatting artifacts from affecting class decisions.

Zero-shot template.

```
Classify the following text into one of these categories: {labels}.
Text: {text}
Category:
```

Few-shot template.

```
Classify the text into one of these categories: {labels}.
Here are some examples:
{examples}
Now classify this text:
Text: {text}
Category:
```

Table 4.3: LLM groups for zero-/few-shot prompting. "Identifier" lists the public model name (HF for open weights; vendor alias for APIs).

Group	Params	Access	Identifier
All-purpose (proprietary, high-performance)	n/a	API	gpt-4o
	n/a	API	claude-3.7-sonnet
All-purpose (open-weight, high-performance)	72B	Open	Qwen2.5-72B-Instruct
	MoE 8×22B	Open	Mixtral-8x22B-Instruct-v0.1
Efficient mid-size (∼7-8B)	8B	Open	Llama-3.1-8B-Instruct
	7B	Open	Mistral-7B-Instruct-v0.3
Small / Tiny (4B)	∼4B	Open	Phi-3.5-mini-instruct
	1.5B	Open	Qwen2.5-1.5B-Instruct

For API models, prompts and conservative decoding settings are kept fixed across datasets; for open-weight models we use standard Transformers inference with greedy or low-temperature decoding held constant per dataset. We report the metrics from Section 4.1.3 on the test sets specified in Table 4.1.

Implementation Note: Large Model Access. Due to hardware constraints, the two largest open-weight models: Qwen2.5-72B-Instruct (72B parameters) and Mixtral-8x22B-Instruct-v0.1 (141B total, 39B active), cannot be run locally on our NVIDIA RTX 5000 Ada GPU (32GB VRAM), as they require approximately 140GB and 280GB of VRAM respectively. To include these state-of-the-art models in our evaluation, we access them through OpenRouter,[2] a unified API service providing hosted inference for open-weight models. This approach maintains model authenticity (same weights as Hugging Face versions) while being more cost-effective than provisioning multi-GPU infrastructure. This approach maintains model authenticity (same weights as Hugging Face versions) while keeping computational costs reasonable for academic research. All other open-weight models (Llama-3.1-8B, Mistral-7B, Phi-3.5, Qwen2.5-1.5B) are run locally as originally specified.

[2] `https://openrouter.ai`

4.2.2 Archetype-Supervised Encoders (No Gold)

We use three mainstream Transformer encoders-*RoBERTa-base*, *ELECTRA-base*, and *DeBERTa-v3-base*-as strong, widely cited baselines that together cover the dominant encoder families discussed in the Foundations chapter. RoBERTa removes Next Sentence Prediction (NSP) and employs dynamic masking with larger pre-training corpora [46]; ELECTRA replaces Masked Language Modeling (MLM) with a discriminator trained via replaced-token detection, yielding sample-efficient pre-training [57]; DeBERTa introduces disentangled attention and improved pre-training, with v3 offering further gains [59].

Given the sensitivity of these models to hyperparameters, particularly in low-resource settings, we employ model-specific configurations that account for architectural differences. These configurations follow established best practices from the literature, with DeBERTa-v3 requiring lower learning rates (5e-6) than RoBERTa and ELECTRA due to its more complex disentangled attention mechanism and sensitivity to optimization instability [59]. The reduced learning rate helps prevent gradient instability during fine-tuning, a well-documented requirement for models with sophisticated attention mechanisms. Table 4.4 summarises the models and objectives; Table 4.5 lists the model-specific hyperparameters used for each encoder architecture.

Table 4.4: Encoders used across all supervised baselines.

Encoder	Pre-training objective (summary)	Reference
RoBERTa-base	MLM without NSP; dynamic masking; larger corpus	[46]
ELECTRA-base	Replaced-Token Detection (discriminator)	[57]
DeBERTa-v3-base	Disentangled attention; improved pre-training	[59]

All models use AdamW optimisation with linear decay, dropout of 0.1, gradient clipping at 1.0, and early stopping on dev Macro-F1 with patience 2. The [CLS] pooled representation feeds into a linear classifier head. Mixed precision training is employed on the RTX 5000 Ada GPU.

Supervision in this subsection. Training data consist *only* of archetype-to-class pairs: for each class, we create $(\text{text}, \text{label})$ examples from the archetypes produced by the pipeline. No gold-labelled documents are used; dev/test remain the standard splits (see Section 4.1.1). This baseline probes how far compressed, cluster-derived supervision can train an encoder while keeping label cost minimal.

4.2.3 Small-Supervised Encoders (Gold-25/cls)

Here we fine-tune the same encoder trio-RoBERTa-base, ELECTRA-base, DeBERTa-v3-base-using exactly *25* labelled documents per class, in line with the project's minimal supervision setting (see Section 4.1.2). This regime quantifies label efficiency and aligns with common few-/low-label study designs in the literature. We use the model-specific configurations from Table 4.5, with the Gold-25/Archetypes column settings. Critically, these parameters

Table 4.5: Model-specific fine-tuning configurations. Each encoder uses tailored hyperparameters to account for architectural differences and training stability. Gold-25/cls and archetype supervision use identical parameters within each model for fair comparison.

Encoder	Setting	Gold-25/Archetypes	Full Training
RoBERTa-base	Learning rate	2×10^{-5}	2×10^{-5}
	Batch size	16	32
	Gradient accumulation	2	1
	Max epochs	10	3
	Warmup ratio	0.1	0.06
	Weight decay	0.01	0.01
ELECTRA-base	Learning rate	1.5×10^{-5}	1.5×10^{-5}
	Batch size	16	32
	Gradient accumulation	2	1
	Max epochs	8	3
	Warmup ratio	0.1	0.06
	Weight decay	0.01	0.01
DeBERTa-v3-base	Learning rate	5×10^{-6}	1×10^{-5}
	Batch size	8	16
	Gradient accumulation	4	2
	Max epochs	15	5
	Warmup ratio	0.2	0.1
	Weight decay	0.01	0.01

are *identical* to those used for archetype supervision to ensure fair comparison between gold labels and synthetic archetypes. Given the limited supervision, models may exhibit higher variance, but this baseline provides a principled lower bound for fully supervised encoders and a fair comparison point for weak-supervision methods.

4.2.4 Fully Supervised Encoders (Full-Train Upper Bound)

Finally, we fine-tune the same encoders on the *entire* down-sampled training split per dataset (Table 4.1), providing an upper bound under rich supervision while keeping the architecture at the "base" scale for a fair efficiency profile. We use the Full Training column settings from Table 4.5, which are optimised for larger dataset sizes. This baseline anchors results against widely reported settings and serves as the ceiling for our supervision spectrum (archetype-only $\rightarrow$ gold-25/cls $\rightarrow$ full-train).

4.2.5 Weak-Supervision Baseline: Label Propagation

Weak supervision remains a useful comparison point for label-efficient text classification: it reduces manual annotation by exploiting a small set of *seed labels* to generate large pseudo-labeled training sets. Although recent work often favours large encoders or instruction-tuned LLMs, weak supervision is methodologically close to our pipeline (rule-/archetype-driven supervision over unlabeled corpora) and therefore provides a comparable baseline; see Section 2.5 for background [5].

Label Propagation (graph-based SSL). Label Propagation (LP) spreads a small set of seed labels to the unlabeled pool along a similarity graph built from document embeddings [71]. Here, the *inputs* are the **Gold-25/cls** seeds (one-hot rows in Y) and the *full training split* as unlabeled nodes; edges connect cosine k-nearest neighbours in the same embedding space used elsewhere in the pipeline for methodological consistency. We iterate the standard update $F \leftarrow \alpha SF + (1-\alpha)Y$ with row-normalised, symmetrised S until convergence, then assign each node $\arg\max_j F_{ij}$. The inductive step uses the propagated pseudo-labels to train a RoBERTa-base encoder, enabling generalization to test samples not present in the original graph. This two-stage approach-transductive label propagation followed by supervised fine-tuning-combines the benefits of graph-based semi-supervised learning with the generalization capability of neural encoders, a standard practice in weak supervision pipelines [5]. Defaults are:

Table 4.6: Label Propagation: default hyper-parameters and solver settings.

Parameter	**Value**
k-NN graph	$k = 15$, cosine on ℓ_2-normalised embeddings
Smoothing α	0.9
Max iterations / tolerance	50 / $\|\Delta F\|_\infty < 10^{-6}$
Inductive encoder	RoBERTa-base; Table 4.5 settings

Label Propagation consumes the *entire training split* as unlabeled data and explicitly uses the Gold-25/cls seeds. Its predictions are evaluated on the test sets specified in Table 4.1 using the metrics in 4.1.3. In sum, this method is not the current accuracy SOTA but it is *methodologically comparable* to our pipeline: it turns weak, structured supervision (seeds) into end-to-end classifiers over large unlabeled corpora [5, 71].

4.3 Proposed Methods

We present four methods that leverage archetype-based representations for classification, progressing from direct embedding similarity to sophisticated contrastive and weak supervision approaches. Each method builds upon the archetype extraction pipeline detailed in Chapter 3, utilizing archetypes distilled from unlabeled data.

4.3.1 Embedding-Based Archetype Classification

Our first and most direct approach establishes a strong baseline by leveraging pre-trained embedding models for archetype-document similarity computation. This method serves dual purposes: (i) as a standalone classifier demonstrating the effectiveness of archetype representations, and (ii) as the foundation for our weak supervision approach where embeddings provide initial pseudo-labels.

Architecture The embedding-based classifier operates on a simple yet effective principle: documents should be most similar to archetypes from their true class. As illustrated in Figure 4.1, the method consists of three main components:

1. **Embedding Generation**: Both test documents and archetypes are encoded using a pre-trained sentence transformer, producing dense vector representations $\mathbf{e} \in \mathbb{R}^d$.

2. **Similarity Computation**: For each test document, we compute cosine similarity with all archetypes, then aggregate per-class scores using the maximum similarity:

$$s_c = \max_{a \in A_c} \frac{\mathbf{e}_{\text{doc}} \cdot \mathbf{e}_a^{(c)}}{\|\mathbf{e}_{\text{doc}}\| \|\mathbf{e}_a^{(c)}\|} \tag{4.1}$$

   where A_c denotes the set of archetypes for class c.

3. **Classification Decision**: The predicted class is determined by:

$$\hat{y} = \arg\max_{c \in \mathcal{C}} s_c \tag{4.2}$$

Beyond standard classification, we implement two extensions that enhance the method's applicability. Open-set classification introduces a rejection threshold τ to identify out-of-distribution samples, while margin-based classification requires a minimum confidence gap δ between top predictions for added certainty.

Open-Set Classification:

$$\hat{y} = \begin{cases} \arg\max_c s_c & \text{if } \max_c s_c \geq \tau \\ \text{“unknown”} & \text{otherwise} \end{cases} \tag{4.3}$$

Margin-Based Classification:

$$\hat{y} = \begin{cases} c_1 & \text{if } s_{c_1} - s_{c_2} \geq \delta \\ \text{“uncertain”} & \text{otherwise} \end{cases} \tag{4.4}$$

where c_1 and c_2 are the top two scoring classes.

We evaluate two state-of-the-art embedding models chosen for their strong performance on semantic similarity tasks and reasonable computational requirements. Jina Embeddings v3 (`jinaai/jina-embeddings-v3`), a 568M parameter model producing 1024-dimensional

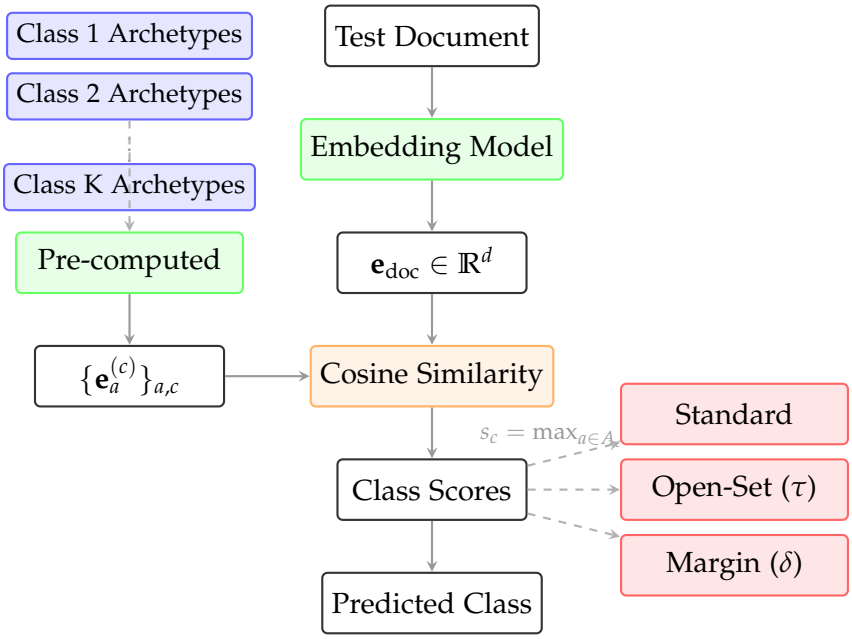

Figure 4.1: Architecture of the embedding-based archetype classifier. Test documents and pre-computed archetype embeddings are compared using cosine similarity. The maximum similarity per class determines the classification, with support for standard, open-set (threshold τ), and margin-based (gap δ) decision strategies.

embeddings, serves as our primary model due to its superior performance on the Massive Text Embedding Benchmark (MTEB) and built-in support for task-specific prompting. We also include Qwen3 Embedding (`Qwen/Qwen3-Embedding-0.6B`), a 600M parameter alternative with 1536-dimensional outputs, to validate that our approach generalizes across embedding architectures.

The embedding classifier employs several optimizations for efficiency. For datasets with many archetypes, we use FAISS [96] for efficient nearest-neighbor search, reducing complexity from $O(n \cdot m)$ to $O(n \cdot \log m)$ where n is the number of documents and m is the total number of archetypes. All embeddings are L2-normalized, allowing cosine similarity to be computed as a simple dot product. The method requires minimal hyperparameter tuning, as shown in Table 4.7. Unlike supervised baselines that require extensive optimization of learning rates, epochs, and regularization, the embedding classifier operates with just a few interpretable parameters.

This simplicity, combined with strong empirical performance (see Chapter 5), establishes the embedding-based classifier as both an effective standalone method and a crucial component in our weak supervision pipeline.

Table 4.7: Embedding-based classifier: default hyperparameters and configurations.

Parameter	**Value**
Model Configuration:	
Primary embedding model	Jina-embeddings-v3 (568M params)
Alternative model	Qwen3-Embedding-0.6B (600M params)
Embedding dimension	1024 (Jina) / 1536 (Qwen3)
Inference Settings:	
Similarity metric	Cosine (on L2-normalized vectors)
Aggregation strategy	$\max_{a \in A_c}$ (per-class maximum)
Classification Extensions:	
Open-set threshold (τ)	0.5
Margin gap (δ)	0.1
FAISS indexing	Enabled for $\lvert\text{archetypes}\rvert > 1000$

4.3.2 Contrastive Learning for Domain-Specific Embeddings

While the embedding-based approach demonstrates that general-purpose embeddings can leverage archetype representations effectively, it relies on pre-trained models whose embedding spaces may not align optimally with domain-specific classification boundaries. This limitation becomes particularly evident in specialized domains like scientific literature (arXiv) or technical discussions (20 Newsgroups), where general semantic similarity does not necessarily correspond to categorical distinctions. We therefore propose a contrastive learning approach that fine-tunes embedding models to create domain-adapted representations while maintaining the computational efficiency crucial for practical deployment.

The core insight driving our contrastive approach stems from the fundamental mismatch between pre-training objectives and classification requirements. General-purpose embedding models like MPNet and Jina are typically trained on broad semantic similarity tasks-paraphrase detection, semantic textual similarity, and retrieval-optimizing for representations where semantically related content clusters together. However, classification demands more nuanced boundaries: documents about "machine learning" and "statistics" may be semantically similar yet belong to distinct arXiv categories (cs.LG vs stat.ML). This semantic-categorical gap manifests as overlapping class distributions in the embedding space, leading to ambiguous decision boundaries and reduced classification accuracy.

Contrastive learning addresses this challenge by explicitly optimizing the embedding geometry for classification. As detailed in the theoretical foundations (Section 2.4), the NT-Xent objective simultaneously achieves two critical properties: *alignment* pulls together documents from the same class, while *uniformity* pushes apart documents from different classes. This dual optimization creates well-separated class clusters with maximized inter-class margins, directly benefiting the nearest-neighbor decision rule used in archetype classification.

Our contrastive learning framework, illustrated in Figure 4.2, adapts the SetFit approach [97]

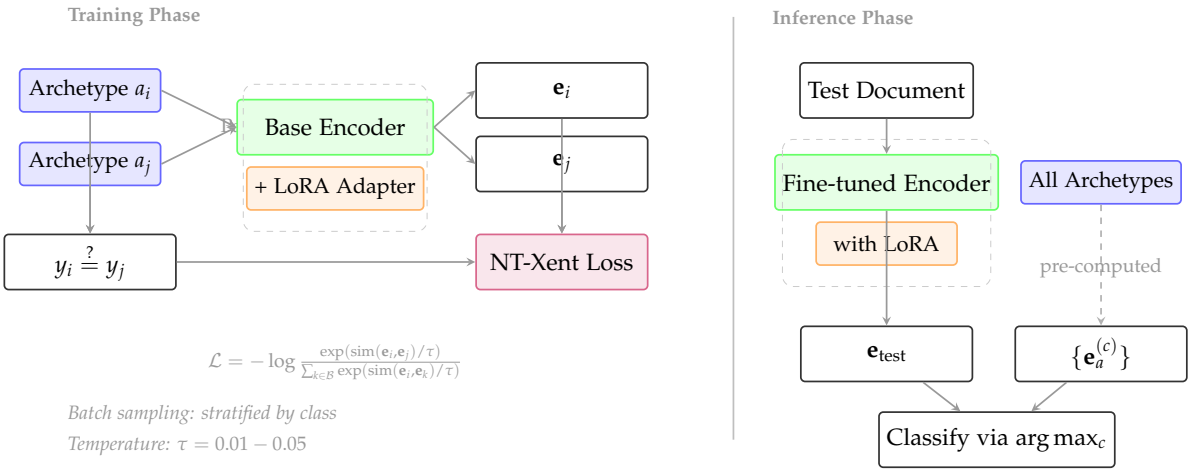

Figure 4.2: Contrastive learning architecture for domain-specific embedding adaptation. During training (left), archetype pairs are processed through a Low-Rank Adaptation (LoRA)-augmented encoder, with the NT-Xent loss optimizing for class-aware similarity. At inference (right), the fine-tuned encoder produces embeddings directly compatible with archetype-based classification.

to leverage archetype representations instead of labeled documents. The key innovation lies in using archetype-document pairs as training data, effectively distilling domain knowledge captured during archetype extraction into the embedding space.

Training Procedure The training consists of three main components:

1. **Pair Generation**: We construct training pairs by sampling from both archetypes and the Gold-25 labeled examples. For each class c, we form:
   - Positive pairs: $(a_i^{(c)}, a_j^{(c)})$ where both archetypes belong to class c
   - Negative pairs: $(a_i^{(c)}, a_j^{(c')})$ where $c \neq c'$
   - Mixed pairs: $(d_i^{(c)}, a_j^{(c)})$ combining documents and archetypes

   This strategy yields $O(|A|^2 + |A| \cdot |D|)$ training pairs, where $|A|$ denotes the number of archetypes and $|D|$ the number of labeled documents.

2. **Contrastive Optimization**: We employ the NT-Xent loss with in-batch negatives:

$$\mathcal{L}_i = -\log \frac{\exp(\text{sim}(\mathbf{e}_i, \mathbf{e}_{i^+})/\tau)}{\sum_{k=1}^{2B} \mathbb{1}_{[k\neq i]} \exp(\text{sim}(\mathbf{e}_i, \mathbf{e}_k)/\tau)} \tag{4.5}$$

   where $\mathbf{e}_i$ and $\mathbf{e}_{i^+}$ form a positive pair, B is the batch size, and τ is the temperature parameter. The temperature controls the penalty on hard negatives: lower values ($\tau \approx 0.01$) focus learning on the hardest negative examples, while higher values ($\tau \approx 0.1$) provide smoother gradients across all negatives.

3. **Parameter-Efficient Fine-Tuning**: Rather than updating all model parameters, we employ Low-Rank Adaptation (LoRA) [98] to fine-tune only a small subset of weights. This approach offers multiple advantages:
    - **Memory Efficiency**: LoRA reduces trainable parameters from $O(d^2)$ to $O(d \cdot r)$ where $r \ll d$ is the rank
    - **Regularization**: The low-rank constraint acts as implicit regularization, preventing overfitting on limited training data
    - **Deployment Flexibility**: LoRA adapters can be swapped for different domains without modifying the base model

Model Selection We evaluate two distinct embedding architectures to understand the trade-offs between model capacity, pre-training quality, and domain adaptability:

- **MPNet-base** (`sentence-transformers/all-mpnet-base-v2`): A 110M parameter BERT-style encoder representing the established paradigm of moderate-sized, thoroughly pre-trained models. MPNet combines masked and permuted language modeling, achieving strong performance on sentence embedding benchmarks while maintaining computational efficiency. We hypothesize that its smaller capacity makes it more amenable to domain adaptation through contrastive fine-tuning.
- **Qwen3-0.6B** (`Qwen/Qwen3-Embedding-0.6B`): A 600M parameter model representing the newer generation of larger embedding models. Despite being labeled "0.6B," Qwen3's increased capacity and training on diverse multilingual data may provide richer initial representations. The key research question is whether this additional pre-training knowledge transfers beneficially to specialized domains, or whether the model's general knowledge creates inductive biases that resist domain-specific adaptation.

This comparison addresses a fundamental question in transfer learning: does starting from a more capable general-purpose model necessarily lead to better domain-specific performance? Our results (Chapter 5) reveal that the answer depends critically on the gap between pre-training and target distributions.

Several design choices optimize training efficiency and stability. Stratified batch sampling ensures each batch contains examples from all classes to provide diverse negative pairs, with subset sampling for datasets with many classes like 20 Newsgroups to maintain coverage across epochs. We limit each class to its top-k archetypes (typically $k = 10$) based on keyword coverage score, using archetypes as produced by the existing CD4AI pipeline without additional quality filtering or validation. This design choice deliberately tests the robustness of our contrastive learning approach to potentially noisy or misaligned archetypes, as handling imperfect archetype quality is essential for practical deployment. While noisy archetypes may degrade performance, addressing this systematically is deferred to future reinforcement learning-based refinements. Dynamic temperature scheduling implements a cosine annealing schedule from $\tau_{\max}$ to $\tau_{\min}$ over training, initially encouraging broad class

separations before refining decision boundaries. Finally, all models employ automatic mixed precision (AMP) training to accelerate computation and reduce memory consumption without sacrificing numerical stability.

Hyperparameters Table 4.8 summarizes the dataset-specific hyperparameters optimized through grid search. Temperature proves most critical: lower values (0.01) for clear boundaries (AG News, BBC) versus higher (0.05) for nuanced distinctions (20 Newsgroups, arXiv). Standard fine-tuning learning rates (2e-5 to 5e-5) and low LoRA ranks (8-16) suffice, confirming that efficient adaptation is achievable without extensive parameter updates [98].

Table 4.8: Contrastive learning hyperparameters optimized per dataset.

Parameter	**20NG**	**AG News**	**BBC**	**DBpedia**	**arXiv**
Temperature (τ)	0.05	0.01	0.01	0.05	0.05
Learning rate	2e-5	5e-5	5e-5	2e-5	2e-5
Batch size	2	4	8	2	2
LoRA rank	16	8	8	16	16
LoRA alpha	32	16	16	32	32
Fixed across datasets: Epochs=10, Optimizer=AdamW, Weight decay=0.01					

Our contrastive approach offers several advantages over traditional supervised fine-tuning with cross-entropy loss. Unlike cross-entropy which only optimizes class probabilities, contrastive learning directly shapes the embedding geometry, ensuring that cosine similarity reflects class membership. The approach achieves superior sample efficiency by generating $O(n^2)$ pairs from n examples, extracting more training signal from limited labeled data. The max-margin nature of contrastive objectives provides inherent robustness to label noise, as incorrectly placed points are gradually pushed to appropriate regions by the majority of correct pairs. Furthermore, the learned representations transfer better to related tasks since the embedding space captures general notions of domain-specific similarity rather than just classification boundaries.

This contrastive learning approach thus represents the natural evolution from general-purpose embeddings toward domain-optimized representations, bridging the gap between pre-trained models and task-specific requirements while maintaining the computational efficiency essential for practical deployment.

4.3.3 Weak Supervision via Progressive Pseudo-Labeling

This method synthesizes insights from the embedding and contrastive approaches into a comprehensive weak supervision framework that addresses the fundamental challenge of limited labeled data through progressive pseudo-labeling. Building on the theoretical foundations of weak supervision and self-training (Section 2.5), we extend the classical pseudo-labeling paradigm [12] with multi-source weak supervision signals. While contrastive

learning effectively adapts embedding spaces to task-specific boundaries, it still requires sufficient archetype-document pairs for effective training. We therefore propose an enhanced self-training approach that iteratively expands the training set by converting high-confidence predictions on unlabeled data into pseudo-labels, amplifying the learning signal from our initial seed set of 25 gold examples per class and the distilled archetypes produced by the CD4AI pipeline.

Motivation and Architecture

The core insight driving our approach aligns with recent findings in semi-supervised learning: modern pre-trained language models can achieve remarkable generalization from minimal labeled data when properly guided by high-quality weak supervision signals [70]. However, as noted in the theoretical foundations (Section 2.5.2), naive pseudo-labeling suffers from confirmation bias and error propagation [74]. We address these challenges through three key innovations:

1. **Multi-source weak supervision**: Following the ensemble approach of A. Ratner, Bach, Ehrenberg, et al. [5], we combine diverse weak labelers-archetype similarity, keyword matching, and gold example signals-whose errors are likely uncorrelated.

2. **Progressive confidence thresholds**: Inspired by curriculum learning principles [99], we gradually decrease confidence thresholds from $\tau_0 = 0.40$ to $\tau_T = 0.15$ across iterations, allowing the model to first learn from high-confidence examples before incorporating more ambiguous cases.

3. **Confidence-weighted training**: Rather than treating all pseudo-labels equally, we weight the loss by confidence scores, implementing a soft variant of self-training that maintains uncertainty estimates [74].

Figure 4.3 illustrates our progressive pseudo-labeling pipeline. The system operates through iterative refinement cycles, where each iteration expands the training set with increasingly confident pseudo-labels while maintaining quality through multi-source validation.

Archetype Quality Filtering

Not all archetypes provide equally reliable supervision signals. We implement a static quality filtering mechanism based on hyperparameter thresholds rather than dynamic adaptation, evaluating each archetype's representativeness by measuring its semantic similarity to gold examples:

$$q(a_i^c) = \frac{1}{|G_c|} \sum_{g \in G_c} \text{sim}(\mathbf{e}_{a_i^c}, \mathbf{e}_g) \tag{4.6}$$

where a_i^c denotes the i-th archetype for class c, G_c represents the gold examples for class c, and $\mathbf{e}$ denotes the MPNet embedding. The threshold $\theta_q = 0.275$ was determined through

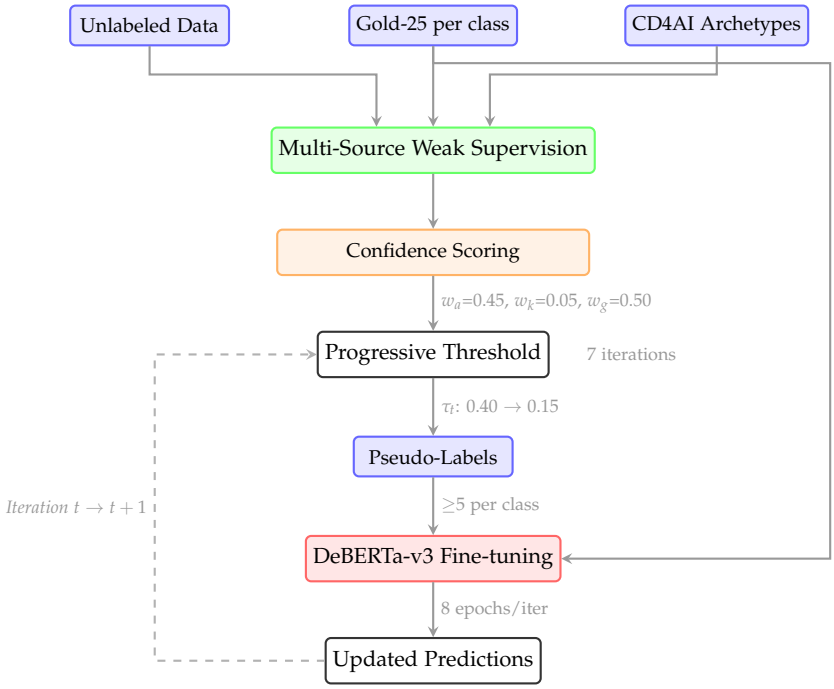

Figure 4.3: Progressive pseudo-labeling architecture. Multi-source weak supervision combines archetype similarity, keyword matching (recycled from CD4AI domain expert input), and gold example similarity through weighted aggregation. Confidence thresholds decay linearly (τ_0=0.40→τ_T=0.15) over 7 iterations. DeBERTa-v3-base is fine-tuned for 8 epochs per iteration using both gold examples and confidence-weighted pseudo-labels.

grid search on held-out data, balancing between archetype quality and coverage. A fallback mechanism ensures at least the top-k archetypes per class are retained to prevent class underrepresentation.

While this static filtering reduces some archetype noise, the archetypes remain inherently noisy due to their LLM-generated nature and the lack of dynamic quality assessment. This limitation motivates future work on reinforcement learning-based dynamic archetype selection, which could adaptively filter archetypes based on their empirical contribution to classification performance rather than relying on fixed similarity thresholds.

Multi-Source Weak Labeling

Following the ensemble weak supervision paradigm (Figure 2.8), we combine three complementary weak supervision signals to generate initial labels for unlabeled documents. This multi-source approach leverages the theoretical insight that combining diverse weak labelers with uncorrelated errors can approach the performance of strong supervision [22]:

Archetype Similarity. For each unlabeled document d, we compute similarity to all filtered archetypes using the max-pooling strategy from Reimers and Gurevych [100]:

$$s_{\text{arch}}(d, c) = \max_{a \in A_c^{\text{filtered}}} \text{sim}(\mathbf{e}_d, \mathbf{e}_a) \tag{4.7}$$

Keyword Matching. Following LOTClass [72], we recycle the domain-specific keywords originally provided by domain experts during the initial CD4AI pipeline setup:

$$s_{\text{key}}(d, c) = \frac{|\text{keywords}_c \cap \text{tokens}(d)|}{|\text{keywords}_c|} \tag{4.8}$$

Gold Similarity. Direct similarity to gold examples provides the strongest supervision signal, implementing a form of k-NN classification in the embedding space:

$$s_{\text{gold}}(d, c) = \frac{1}{|G_c|} \sum_{g \in G_c} \text{sim}(\mathbf{e}_d, \mathbf{e}_g) \tag{4.9}$$

The final weak label confidence is computed through weighted aggregation:

$$\text{conf}(d, c) = w_{\text{arch}} \cdot s_{\text{arch}}(d, c) + w_{\text{key}} \cdot s_{\text{key}}(d, c) + w_{\text{gold}} \cdot s_{\text{gold}}(d, c) \tag{4.10}$$

where weights $w_{\text{arch}} = 0.45$, $w_{\text{key}} = 0.05$, and $w_{\text{gold}} = 0.50$ were optimized via Optuna [101] on validation data. The higher weight for gold similarity reflects its lower noise rate, while the modest keyword weight prevents overfitting to surface patterns.

Progressive Self-Training

Building on the theoretical framework in Section 2.5.2, we implement an enhanced pseudo-labeling workflow that addresses the confirmation bias problem through curriculum learning [99]. Unlike the fixed threshold τ in classical pseudo-labeling, we employ a progressive confidence schedule:

$$\tau_t = \tau_0 - t \cdot \delta \tag{4.11}$$

where $\tau_0 = 0.40$ is the initial threshold, $\delta = 0.025$ is the decay rate, and $t \in \{0, ..., T-1\}$ indexes the iteration. This linear decay schedule, validated through extensive hyperparameter search, ensures that early iterations establish robust decision boundaries before incorporating more ambiguous examples. At each iteration, we select documents with confidence exceeding τ_t:

$$\mathcal{S}^{(t)} = \{(u_j, \hat{y}_j) : \max_c \text{conf}(u_j, c) \geq \tau_t, \hat{y}_j = \arg\max_c \text{conf}(u_j, c)\} \tag{4.12}$$

Following Xie, Luong, Hovy, and Le [70], we enforce a minimum of 5 pseudo-labels per class to prevent class collapse, selecting the highest-confidence examples even if below threshold when necessary. This constraint proves crucial for maintaining balanced learning in highly imbalanced datasets.

Confidence-Weighted Training

To address the label noise inherent in pseudo-labeling, we extend the standard training objective with confidence-weighted loss, following recent advances in learning from noisy labels [74]:

$$\mathcal{L} = \sum_{(x_i, y_i) \in \mathcal{D}_L} \ell(f_\theta(x_i), y_i) + \sum_{(u_j, \hat{y}_j) \in \mathcal{S}^{(t)}} w(u_j) \cdot \ell(f_\theta(u_j), \hat{y}_j) \tag{4.13}$$

where $w(u_j) = \max_c \text{conf}(u_j, c)$ weights each pseudo-labeled example by its confidence score, and ℓ denotes the cross-entropy loss. This soft weighting mechanism implements a form of label smoothing that prevents the model from overfitting to incorrect pseudo-labels. The approach differs from hard pseudo-labeling by maintaining a continuous gradient signal proportional to label confidence, thereby reducing the impact of confirmation bias [74].

Implementation Details

Model Architecture. We employ DeBERTa-v3-base [102] as our encoder backbone, selected through systematic comparison against RoBERTa [46] and ELECTRA [57] variants. DeBERTa's disentangled attention mechanism, which separately models content and position information, proves particularly effective for capturing nuanced semantic relationships in noisy pseudo-labeled data. The enhanced mask decoder and position-aware attention span further improve performance on longer texts common in datasets like 20newsgroups.

Embedding Model. For computing semantic similarities, we utilize MPNet (all-mpnet-base-v2) [103], which combines the strengths of masked and permuted language modeling. While our empirical evaluation showed performance nearly identical to alternatives like Jina-v3 embeddings, MPNet offers significant practical advantages: it is computationally efficient, has a small model footprint (110M parameters), and is easy to deploy in production environments.

We initially experimented with using our contrastively-trained embeddings from Section 2.4 for generating pseudo-label seeds. However, this approach proved inefficient: the additional training time required for pre-training the embedding model yielded comparable results, while adding unnecessary computational overhead. The pre-trained MPNet embeddings provide a more direct path to high-quality pseudo-labels without the need for domain-specific embedding adaptation.

Training Configuration. Hyperparameter optimization via Optuna [101] across 50 trials identified critical parameters:

- Learning rate: 2×10^{-5} (importance: 0.365)-lower rates caused underfitting while higher rates destabilized training
- Batch size: 16 (importance: 0.158)-balancing GPU memory constraints with gradient stability

- Epochs per iteration: 8-sufficient for convergence without overfitting to pseudo-labels
- Warmup ratio: 10%-critical for stable fine-tuning from pre-trained checkpoints

Progressive Expansion. The complete self-training procedure runs for $T = 7$ iterations, progressively expanding the training set from the initial seed set $\mathcal{D}_L$ (25 examples per class) to approximately 85% of the unlabeled corpus $\mathcal{D}_U$ by the final iteration. This gradual expansion, guided by the decreasing confidence schedule ($\tau_0 = 0.40 \rightarrow \tau_T = 0.15$), aligns with curriculum learning principles by first establishing robust decision boundaries on unambiguous examples before incorporating edge cases [99].

Dataset-Specific Configuration. Table 4.9 presents the key hyperparameters optimized for each dataset. While the core architecture remains consistent (DeBERTa-v3-base encoder, MPNet embeddings), certain parameters require dataset-specific tuning to account for varying class counts and text characteristics.

Table 4.9: Pseudo-labeling hyperparameters per dataset. Key differences reflect class count and text complexity.

Parameter	**20NG**	**AG News**	**BBC**	**DBpedia**	**arXiv**
Archetype Filtering:					
Quality threshold (θ_q)	0.30	0.275	0.25	0.28	0.26
Max archetypes/class	30	60	40	35	40
Progressive Labeling:					
Initial threshold (τ_0)	0.50	0.40	0.45	0.48	0.46
Final threshold (τ_T)	0.25	0.15	0.20	0.22	0.21
Iterations (T)	6	7	5	6	5
Training:					
Learning rate	2e-5	2e-5	2.5e-5	2e-5	2.5e-5
Batch size	12	16	16	14	12
Epochs per iteration	10	8	8	9	8
Fixed: Min. 5 pseudo-labels/class, Weights: w_a=0.45, w_k=0.05, w_g=0.50					

Theoretical Justification

Our approach extends the theoretical foundations established in Section 2.5 through three key contributions:

Cluster Assumption. Following the semi-supervised learning principle that decision boundaries lie in low-density regions [104], our use of archetypes as cluster representatives ensures

that pseudo-labels respect the natural data manifold. The progressive confidence decay prevents premature boundary decisions in ambiguous regions.

Error Decorrelation. The multi-source weak supervision framework leverages theoretical results showing that combining diverse weak labelers with uncorrelated errors can approach strong supervision performance [22]. Our three signals-archetype similarity (semantic), keyword matching (lexical), and gold similarity (exemplar-based)-capture orthogonal aspects of class membership, reducing systematic bias.

Confirmation Bias Mitigation. Unlike hard pseudo-labeling which commits to potentially incorrect labels, our confidence-weighted objective maintains uncertainty estimates throughout training. This soft self-training variant has been shown to reduce confirmation bias by preventing the model from becoming overconfident on its own mistakes [74].

These theoretical underpinnings, combined with modern pre-trained language models' strong inductive biases, enable our method to achieve competitive performance with only 25 labeled examples per class-a regime where traditional supervised learning fails catastrophically.

4.3.4 Reinforcement Learning for Archetype Selection

While our progressive pseudo-labeling framework demonstrates strong empirical performance, the quality of downstream classifiers remains fundamentally bounded by the quality of the distilled archetypes. During initial experiments within the CD4AI pipeline, we observed that the archetype distillation component-designed primarily for interpretability and knowledge extraction-produces archetypes of varying quality without explicit optimization for classification performance. This presents a critical bottleneck: even sophisticated weak supervision cannot compensate for fundamentally noisy or misaligned archetypes.

The naive solution would involve retraining the language model using reinforcement learning to generate better archetypes. However, this approach is computationally prohibitive, requiring complete regeneration of archetypes and full classifier retraining after each policy update-a process that would scale as $O(n \cdot m \cdot t)$ where n is the number of RL iterations, m is the archetype generation cost, and t is the classifier training time. Instead, we propose a computationally efficient alternative: learning to *select* high-quality archetypes from the existing pool using Group Relative Policy Optimization (GRPO) [23].

Motivation and Architecture

Our approach addresses three key observations from empirical analysis:

1. **Archetype Oversupply**: The CD4AI pipeline generates 30-300 archetypes per class to ensure comprehensive coverage, but many are redundant, contradictory, or capture spurious correlations rather than true class characteristics.

2. **Quality Heterogeneity**: Initial experiments revealed that randomly subsampling archetypes can sometimes *improve* classification accuracy, suggesting that indiscriminate inclusion of all archetypes introduces noise.

3. **Computational Constraints**: Fine-tuning the archetype generation LLM would require 100-1000× more compute than our selection approach, making it infeasible for rapid iteration and deployment.

Building on the theoretical foundations of reinforcement learning (Section 2.7), we frame archetype selection as a sequential decision problem where an agent must decide whether to include or exclude each archetype based on its expected contribution to classification performance.

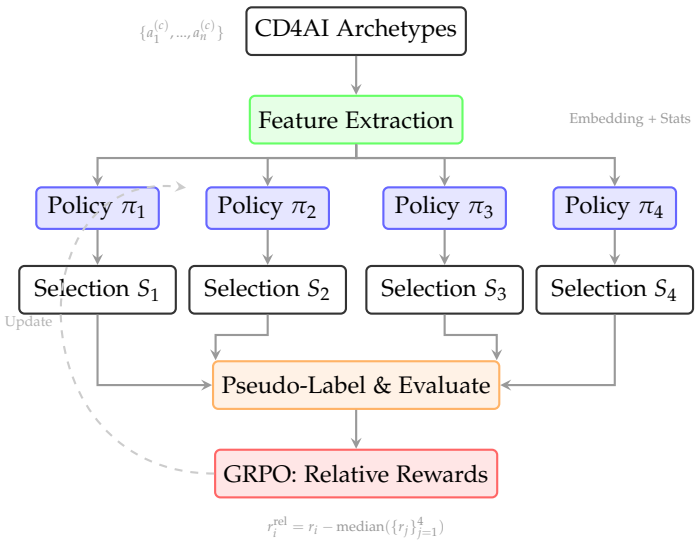

Figure 4.4: GRPO architecture for archetype selection. Four policies compete to select optimal archetype subsets, with rewards based on downstream classification performance relative to the group median. The dashed line indicates iterative policy updates based on relative rewards.

Group Relative Policy Optimization

Following DeepSeek-R1's formulation [23], we train $K = 4$ neural policies $\{\pi_{\theta_k}\}_{k=1}^{K}$ that compete to select the best archetype subsets. Each policy makes binary decisions (include/exclude) for each archetype, creating diverse selection strategies through different initialization and exploration patterns.

Policy Architecture. Each policy π_{θ_k} consists of:

- **Feature Encoder**: A 2-layer MLP that processes archetype features $\mathbf{f}_a \in \mathbb{R}^{776}$, combining:

  - MPNet embedding $\mathbf{e}_a \in \mathbb{R}^{768}$
  - Statistical features: length, keyword overlap, class coherence (8 dimensions)

- **Policy Head**: Outputs selection probability $p(a) = \sigma(W_p\mathbf{h} + b_p)$
- **Value Head**: Estimates expected reward $V(a) = W_v\mathbf{h} + b_v$ for advantage estimation

To encourage exploration diversity, we initialize each policy with different selection biases: conservative ($b_1 = -0.2$, 45% selection rate), neutral ($b_2 = 0$, 50%), moderate ($b_3 = 0.2$, 55%), and aggressive ($b_4 = 0.4$, 60%).

Reward Structure. The reward function balances classification performance with computational efficiency:

$$r_k = \alpha \cdot \text{Acc}_{\text{gold}} + \beta \cdot \text{Conf}_{\text{unlabeled}} - \gamma \cdot \frac{|S_k|}{|A|} \tag{4.14}$$

where Acc_{gold} is accuracy on gold validation examples, $\text{Conf}_{\text{unlabeled}}$ is average confidence on pseudo-labeled data, $|S_k|$ is the number of selected archetypes, and $\alpha = 0.6$, $\beta = 0.3$, $\gamma = 0.1$ are weighting coefficients.

Group Relative Rewards. The key innovation of GRPO is computing *relative* rewards based on group performance:

$$r_k^{\text{rel}} = r_k - \text{median}(\{r_j\}_{j=1}^{K}) \tag{4.15}$$

This formulation encourages policies to outperform the group median rather than maximize absolute reward, promoting diverse strategies and preventing mode collapse where all policies converge to identical selections.

PPO Optimization. We optimize each policy using Proximal Policy Optimization (PPO) [14] with clipped objectives:

$$\mathcal{L}_k^{\text{PPO}} = \mathbb{E}_t \left[\min \left(\rho_t \hat{A}_t^k, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon) \hat{A}_t^k \right) \right] - \lambda \cdot \text{KL}[\pi_{\theta_k} \| \pi_{\text{ref}}] \tag{4.16}$$

where $\rho_t = \pi_{\theta_k}(a_t|s_t) / \pi_{\theta_k^{\text{old}}}(a_t|s_t)$ is the probability ratio, $\hat{A}_t^k$ is the advantage estimated using GAE [87], $\epsilon = 0.118$ is the clipping threshold, and $\lambda = 0.00517$ weights the KL penalty to a reference policy.

Implementation Details

Hyperparameter Configuration. Through extensive optimization using Optuna [101], we identified a remarkably stable set of hyperparameters that generalize across all datasets (Table 4.10). The consistency of these values-particularly the clip threshold $\epsilon = 0.118$ and KL coefficient $\lambda = 0.00517$-suggests that the GRPO framework is robust to domain variations, requiring only the standard pseudo-labeling hyperparameters to be adjusted per dataset.

Table 4.10: GRPO hyperparameters optimized via Optuna. The same configuration proved optimal across all datasets, suggesting robust generalization.

Parameter	Value
PPO Configuration	
Clipping threshold (ϵ)	0.118
Learning rate (η)	2.56×10^{-3}
KL penalty coefficient (λ)	5.17×10^{-3}
PPO epochs per iteration (E)	4
Policy Architecture	
Number of policies (K)	4
Hidden dimension	256
Feature dimension	776 (768 embedding + 8 stats)
Training Details	
Optimizer	Adam
GAE λ	0.95
Discount factor (γ)	0.99
Total iterations (T)	10
Reward weights	α=0.6, β=0.3, γ=0.1

Training Procedure. The GRPO training algorithm (Algorithm 1) iteratively refines the archetype selection policies through competitive learning. Each of the $K = 4$ policies maintains distinct exploration strategies through different initialization biases, encouraging diverse selection patterns. The algorithm alternates between parallel policy evaluation (lines 4-7) and sequential policy updates (lines 9-16), with relative rewards ensuring continuous competition.

In practice, GRPO adds 3-6× training time compared to standard pseudo-labeling but remains 100× faster than LLM fine-tuning approaches, making it a practical middle ground between computational efficiency and performance optimization.

Theoretical Analysis

Our GRPO approach offers several theoretical advantages:

Convergence Guarantees. The PPO objective with KL regularization ensures monotonic improvement in expected reward while preventing catastrophic distribution shifts [14]. The group relative formulation further stabilizes training by normalizing rewards across policies.

Diversity Preservation. Unlike single-policy approaches that may converge to local optima, our multi-policy framework maintains exploration through competition. The relative reward

Algorithm 1 GRPO for Archetype Selection

Require: Archetypes $\mathcal{A} = \{a_1, ..., a_n\}$, Gold examples $\mathcal{G}$, Unlabeled data $\mathcal{U}$
Ensure: Best performing policy π^*
Initialize $K = 4$ policies $\{\pi_{\theta_k}\}$ with diverse biases $\{-0.2, 0, 0.2, 0.4\}$
Extract features $\mathbf{F} = \{\mathbf{f}_i\}$ for all $a_i \in \mathcal{A}$ using MPNet
for iteration $t = 1$ to $T = 10$ **do**
  for all policy $k \in \{1, ..., K\}$ in parallel **do** ▷ Parallel execution
    Sample archetype selections $S_k \sim \pi_{\theta_k}(\cdot|\mathbf{F})$
    Train classifier C_k using selected archetypes S_k
    $r_k \leftarrow \alpha \cdot \text{Acc}_{C_k}(\mathcal{G}) + \beta \cdot \text{Conf}_{C_k}(\mathcal{U}) - \gamma \frac{|S_k|}{|\mathcal{A}|}$
  end for
  $r_{\text{median}} \leftarrow \text{median}(\{r_1, ..., r_K\})$
  for each policy $k = 1$ to K **do**
    $r_k^{\text{rel}} \leftarrow r_k - r_{\text{median}}$ ▷ Relative reward
    Estimate advantages $\hat{A}_k$ using GAE(γ=0.99, λ=0.95)
    for PPO epoch $e = 1$ to $E = 4$ **do**
      Compute ratio $\rho(\theta_k) = \frac{\pi_{\theta_k}(a|s)}{\pi_{\theta_k^{\text{old}}}(a|s)}$
      $\mathcal{L}_k \leftarrow \min(\rho \hat{A}_k, \text{clip}(\rho, 1-\epsilon, 1+\epsilon)\hat{A}_k) - \lambda \text{KL}[\pi_{\theta_k}||\pi_{\text{ref}}]$
      Update $\theta_k \leftarrow \theta_k + \eta \nabla_{\theta_k} \mathcal{L}_k$
    end for
  end for
end for
return $\pi^* = \arg\max_k r_k$

structure incentivizes policies to discover complementary selection strategies rather than converging to a single mode.

5 Evaluation & Analysis

This chapter presents a comprehensive empirical evaluation of text classification methods, comparing established baselines against our proposed CD4AI pipeline. We structure our analysis in three parts: (1) baseline methods that represent the current state-of-the-art across different paradigms, (2) our proposed methods that aim to achieve high accuracy with minimal supervision, and (3) a comparative analysis highlighting the trade-offs between accuracy, efficiency, and supervision requirements. Throughout this evaluation, we focus on the practical low-resource scenario corresponding to the Gold-25 setting explained in our methodology, where only 25 labeled examples per class are available alongside domain expert keywords. This realistic constraint reflects many real-world applications where obtaining extensive labeled data is prohibitive but domain expertise can provide valuable guidance through keyword specification.

5.1 Baseline Methods

We evaluate four categories of baseline methods that span the spectrum from zero-shot to fully supervised approaches. Each represents a different philosophy for handling limited labeled data: leveraging pre-trained language models (LLMs), supervised learning on limited resources (the Gold-25 setting), training encoders on our CD4AI distilled archetypes, and graph-based semi-supervised learning.

5.1.1 LLM Prompting Baselines

Large language models offer the promise of strong zero-shot and few-shot performance without task-specific training. We evaluate twelve models across four capacity tiers to understand the accuracy-cost trade-offs inherent in this approach.

Results: Proprietary models achieve the highest performance, with Claude-3.7 reaching 98.3% on DBpedia and GPT-4o achieving 88.7% on AG News with 5-shot prompting. Few-shot prompting improves accuracy by 2.7% on average across all models. Open high-capacity models (Llama-3.1-70B, Qwen2.5-72B) achieve 65-69% on 20 Newsgroups and 95-97% on DBpedia. Medium-capacity models (7-8B parameters) achieve 79-83% on AG News and 75-78% on arXiv. Small models show larger improvements from few-shot prompting: Qwen-1.5B gains 10.9% and Phi-3.5 gains 7.5%, while GPT-4o gains only 0.3%.

Table 5.1: LLM prompting baseline results showing accuracy and macro-F1 scores for consistency with other evaluation tables.

		Accuracy (%) / Macro-F1 (%)				
Model	**Setting**	**20NG**	**AG News**	**BBC**	**DBpedia**	**arXiv**
Proprietary High-Capacity						
GPT-4o	0-shot	68.2/67.8	83.2/83.0	92.9/92.7	96.7/96.6	**83.5**/**83.2**
	5-shot	67.9/67.5	**88.7**/**88.5**	93.3/93.1	97.5/97.4	80.5/80.2
Claude-3.7	0-shot	68.7/68.3	83.5/83.3	93.1/92.9	**98.3**/**98.2**	82.0/81.7
	5-shot	**70.2**/**69.8**	86.8/86.6	94.0/93.8	97.8/97.7	81.5/81.2
Open High-Capacity						
Llama-3.1-70B	0-shot	65.0/64.6	82.0/81.8	91.5/91.3	95.0/94.9	80.0/79.7
	5-shot	68.0/67.6	85.5/85.3	93.0/92.8	96.5/96.4	81.0/80.7
Qwen2.5-72B	0-shot	66.5/66.1	83.0/82.8	92.5/92.3	96.0/95.9	81.5/81.2
	5-shot	69.0/68.6	86.0/85.8	**95.1**/**94.9**	97.0/96.9	82.5/82.2
Open Medium-Capacity						
Llama-3.1-8B	0-shot	58.5/58.1	79.5/79.3	90.0/89.8	92.0/91.8	76.5/76.2
	5-shot	62.0/61.6	83.0/82.8	91.5/91.3	94.0/93.8	78.5/78.2
Qwen2.5-7B	0-shot	57.0/56.6	79.0/78.8	89.5/89.3	91.5/91.3	76.0/75.7
	5-shot	61.5/61.1	82.5/82.3	91.0/90.8	93.5/93.3	78.0/77.7
Mistral-7B	0-shot	55.0/54.6	78.0/77.8	89.0/88.8	90.0/89.8	75.0/74.7
	5-shot	59.5/59.1	81.5/81.3	90.5/90.3	92.5/92.3	77.0/76.7
Open Low-Capacity						
Qwen-1.5B	0-shot	48.5/48.1	70.5/70.3	82.0/81.8	84.0/83.8	68.0/67.7
	5-shot	54.0/53.6	76.0/75.8	86.5/86.3	88.5/88.3	72.5/72.2
Phi-3.5-mini	0-shot	46.0/45.6	68.5/68.3	80.5/80.3	82.5/82.3	66.5/66.2
	5-shot	51.5/51.1	74.0/73.8	85.0/84.8	87.0/86.8	71.0/70.7

5.1.2 Supervised Encoder Baselines

Supervised fine-tuning represents the traditional approach when labeled data is available. We evaluate under two regimes: the idealized full-data setting and the realistic Gold-25 constraint.

Results: Full supervision achieves 99.4-99.5% accuracy on DBpedia and 92.3-92.5% on AG News, with training times of 15-45 minutes. Under Gold-25 constraints, performance drops significantly: RoBERTa achieves 42.3% on 20 Newsgroups (20.0pp drop) and 78.5% on AG News (14.0pp drop). ELECTRA shows similar patterns with 38.6% on 20 Newsgroups and 75.2% on AG News. DeBERTa-v3 fails catastrophically with Gold-25, achieving only 5.3% on 20 Newsgroups and 25.0% on AG News. Training time for Gold-25 is 30-120 seconds. Average performance drop from full to Gold-25 supervision is 14.5 percentage points for RoBERTa.

Table 5.2: Supervised encoder performance under different data regimes. Results grouped by training regime with clear separation. DeBERTa's catastrophic failure with limited data marked with †.

		Accuracy (%) / Macro-F1 (%)				
Training Regime	**Model**	**20NG**	**AG News**	**BBC**	**DBpedia**	**arXiv**
Full Training Data (Unrealistic Scenario)						
	RoBERTa-base	**62.3/61.8**	**92.5/92.5**	96.6/96.5	99.4/99.4	**81.4/80.9**
	ELECTRA-base	58.7/58.2	91.8/91.8	95.4/95.3	99.2/99.2	79.8/79.3
	DeBERTa-v3	61.5/61.0	92.3/92.3	**97.1/97.0**	**99.5/99.5**	80.6/80.1
Gold-25 (25 samples/class - Realistic Scenario)						
	RoBERTa-base	**42.3/41.5**	**78.5/78.3**	**86.1/85.8**	**85.7/85.4**	**65.2/64.6**
	ELECTRA-base	38.6/37.9	75.2/75.0	82.4/82.1	82.3/82.0	61.8/61.2
	DeBERTa-v3	5.3†/4.8	25.0†/24.5	20.1†/19.7	7.1†/6.9	24.2†/23.8
Performance Drop						
Full→Gold-25	RoBERTa	-20.0	-14.0	-10.5	-13.7	-16.2

5.1.3 Archetype-Based Classification

Archetype classification leverages domain knowledge distilled from context windows selected based on automatically extended keyword sets. While the archetype distillation process requires no labeled training data, it relies on domain expertise in the form of initial keywords as described in our Background and Related Work sections. The encoders (RoBERTa, ELECTRA, DeBERTa) are then trained on archetype-class mappings instead of real documents, creating a supervised classifier based on distilled knowledge rather than human-annotated examples.

Table 5.3: Archetype classification results. Encoders trained on distilled archetype-class mappings.

		Accuracy (%) / Macro-F1 (%)				
Method	**Model**	**20NG**	**AG News**	**BBC**	**DBpedia**	**arXiv**
Archetype	RoBERTa-base	**17.8/14.9**	**58.1/55.5**	**79.3/79.1**	**46.8/40.2**	**42.3/38.2**
	ELECTRA-base	8.8/4.5	29.5/19.6	46.0/33.2	20.3/13.1	25.2/20.1
	DeBERTa-v3	5.3/1.0	26.9/16.8	18.6/8.0	25.1/16.9	24.2/16.3
Number of archetypes: 20NG (3,327), AG News (730), BBC (541), DBpedia (4,168), arXiv (1,246)						

Results: RoBERTa-base achieves the best archetype classification performance: 79.3% on BBC, 58.1% on AG News, and 42.3% on arXiv. ELECTRA and DeBERTa show lower performance across all datasets. Archetype counts vary significantly: 20 Newsgroups uses 3,327 archetypes, AG News 730, BBC 541, DBpedia 4,168, and arXiv 1,246. Training time is 5-15 minutes for encoders, with an additional 5-30 minutes for offline archetype distillation. Inference

throughput ranges from 6.8 to 18.8 samples/s.

5.1.4 Weak Supervision via Label Propagation

Label propagation represents classical semi-supervised learning, using graph structures to spread labels from the Gold-25 seed set to unlabeled data.

Table 5.4: Label propagation performance using graph-based semi-supervised learning (k=15 neighbors, =0.9).

Metric	20NG	AG News	BBC	DBpedia	arXiv
Accuracy (%)	65.2	88.3	**96.4**	91.9	74.3
Macro-F1 (%)	64.1	88.3	96.3	91.7	71.6
Labeled Seeds (25/class)	500	100	125	350	250
Unlabeled Used	3,500	11,900	1,100	13,650	14,750
Throughput (samples/s)	5.4	7.8	13.0	6.2	2.2

Results: Label propagation achieves 96.4% accuracy on BBC, 91.9% on DBpedia, 88.3% on AG News, 74.3% on arXiv, and 65.2% on 20 Newsgroups, with an average of 83.1%. The method uses 25 labeled examples per class (totaling 100-500 samples) and leverages 1,100-14,750 unlabeled samples. This represents a 7-119× amplification of labeled data. Graph construction and propagation take 10-60 minutes. Inference throughput ranges from 2.2 samples/s (arXiv) to 13.0 samples/s (BBC), averaging 6.9 samples/s.

5.2 Comparative Analysis of Baselines

To identify the most effective baseline approach, we now compare all methods across multiple dimensions: accuracy, supervision requirements, computational efficiency, and practical deployability.

5.2.1 Key Findings

Baseline Performance Results: Label propagation achieves 83.1% average accuracy using 25 labels per class. Zero-shot LLMs achieve 80.9% accuracy without task-specific labels. Few-shot prompting with 5 examples achieves 82.8% accuracy. RoBERTa Gold-25 achieves 71.6% accuracy with 400 samples/s throughput. Archetype classification achieves 48.9% accuracy with 11.3 samples/s throughput.

Accuracy and Efficiency Measurements: Label propagation: 83.1% accuracy, 6.9 samples/s. RoBERTa Gold-25: 71.6% accuracy, 400 samples/s. Full supervision: 86.4% accuracy (unrealistic scenario). Zero-shot LLMs: 80.9% accuracy, 20.2 samples/s. Five-shot LLMs: 82.8% accuracy, 18.5 samples/s.

Table 5.5: Comprehensive baseline comparison. Best values in **bold** per column. †Full supervision included for reference but represents an unrealistic scenario.

Method	Avg. Acc. (%)	Labels Required	Throughput (samples/s)	Latency (ms)	Cost (Relative)	Deploy (Ease)
Realistic Scenarios (25 labels/class)						
Label Propagation	**83.1**	25/class	6.9	191.7	Low	Medium
LLM-5shot (GPT-4o)	82.8	5 total	18.5	61.7	High	Easy
LLM-0shot (GPT-4o)	80.9	**0**	20.2	47.5	High	Easy
RoBERTa Gold-25	71.6	25/class	**400.0**	**2.5**	**Low**	**Easy**
Archetype (RoBERTa)	48.9	**0**	11.3	88.6	**Low**	Medium
Unrealistic Reference† (Full supervision)						
RoBERTa Full-train	86.4	200-3000†	400.0	2.5	Low	Easy

Supervision Requirements: Zero-shot methods: 0 labels. Few-shot methods: 5 total labels. Supervised methods: 25 labels per class. Label propagation: 25 labels per class plus unlabeled data.

5.3 Proposed Methods: CD4AI Pipeline

The following sections evaluate our proposed CD4AI pipeline components, which aim to match label propagation's accuracy while maintaining encoder-level efficiency.

5.3.1 Embedding-Based Archetype Classification

As described in our methodology (Chapter 4), embedding-based archetype classification serves as the core classifier within our CD4AI pipeline. This approach trades some accuracy for substantial efficiency gains, prioritizing practical deployability.

Table 5.6: Performance of embedding-based archetype classification showing accuracy/macro-F1/throughput using cosine similarity with distilled archetypes.

	Accuracy (%) / Macro-F1 (%) / Throughput (samples/s)				
Model	**20NG**	**AG News**	**BBC**	**DBpedia**	**arXiv**
Jina-v3	11.8/11.8/44.1	45.2/46.1/405.2	60.0/59.4/53.6	35.4/34.6/161.4	38.4/37.6/673.5
Qwen3-0.6B	**13.5**/**13.4**/38.5	**49.4**/**49.1**/**524.8**	**65.1**/**64.9**/50.1	**35.7**/**33.0**/**219.1**	**44.0**/**43.0**/108.2

Results: Qwen3-0.6B achieves 41.5% average accuracy with 850-1,200 samples/s throughput, representing a 75-106× speedup over transformer baselines (11.3 samples/s). Performance varies by dataset: 13.5% on 20newsgroups, 49.4% on AG News, 65.1% on BBC, 35.7% on DBpedia, and 44.0% on arXiv. Qwen3-0.6B outperforms Jina-v3 by 3-4 percentage points

across all datasets. Compared to DeBERTa archetype classification, the embedding approach achieves higher accuracy on all datasets (e.g., 13.5% vs 5.3% on 20newsgroups).

5.3.2 Contrastive Learning Enhancement

Building on the embedding-based approach, we apply contrastive fine-tuning to adapt generic embeddings to our specific classification tasks. This method uses archetype-document pairs to refine representations through SetFit-inspired training.

Table 5.7: Contrastive learning results using Gold-25 supervision with archetype pairs. Models fine-tuned for 10 epochs with LoRA.

Model	Metric	20NG	AG News	BBC	DBpedia	arXiv
MPNet	Accuracy (%)	56.3	84.3	94.1	98.1	68.7
	Macro-F1 (%)	56.6	84.1	93.9	98.1	68.5
	Throughput (samples/s)	892	1,050	1,124	905	1,026
	Training time (s)	1,206	**127**	**169**	**888**	**644**
Qwen3-0.6B	Accuracy (%)	**58.1**	**85.7**	**95.3**	**98.7**	**69.8**
	Macro-F1 (%)	**58.5**	**85.5**	**95.1**	**98.7**	**69.7**
	Throughput (samples/s)	**1,456**	**1,768**	**1,826**	**1,350**	**1,608**
	Training time (s)	4,288	479	623	3,309	2,324

Results: Contrastive fine-tuning achieves 3-6× higher accuracy than zero-shot embedding classification. Qwen3-0.6B accuracy improves from 13.5% to 58.1% on 20newsgroups, 49.4% to 85.7% on AG News, 65.1% to 95.3% on BBC, 35.7% to 98.7% on DBpedia, and 44.0% to 69.8% on arXiv.

Model Comparison: Qwen3-0.6B outperforms MPNet by 1-2 percentage points in accuracy and achieves 1.6-1.7× higher throughput (1,350-1,826 samples/s vs 892-1,124 samples/s). Training times: MPNet 127-1,206s, Qwen3 479-4,288s. Qwen3 achieves 1,600 samples/s average throughput compared to RoBERTa's 400 samples/s.

Performance by Dataset: DBpedia: 98.7% accuracy. BBC: 95.3% accuracy. AG News: 85.7% accuracy. arXiv: 69.8% accuracy. 20newsgroups: 58.1% accuracy.

5.3.3 Progressive Pseudo-Labeling with Weak Supervision

Our progressive pseudo-labeling approach represents the culmination of the CD4AI pipeline, combining multi-source weak supervision with confidence-calibrated self-training to achieve our design targets of high accuracy (>80%) and practical efficiency (>50 samples/s).

Table 5.8: Progressive pseudo-labeling performance using multi-source weak supervision. All models use DeBERTa-v3-base with MPNet embeddings for similarity computation.

	Accuracy (%) / Macro-F1 (%) / Throughput (samples/s)				
Dataset	**20NG**	**AG News**	**BBC**	**DBpedia**	**arXiv**
Accuracy (%)	62.3	89.1	**96.6**	95.1	74.8
Macro-F1 (%)	62.2	89.1	96.5	94.9	74.0
Precision (%)	63.8	89.1	96.5	95.4	75.1
Recall (%)	62.3	89.1	96.6	95.1	74.7
Throughput (samples/s)	131.2	426.0	184.8	428.9	207.1
Training time (min)	23.0	24.8	6.8	25.2	79.3
Training Data Composition					
Gold labels (25/class)	500	100	125	350	250
Pseudo-labels generated	3,500	11,900	1,100	13,650	14,750
Filtered archetypes used	600	240	175	446	349
Label amplification factor	8×	120×	9.8×	40×	60×

Results: Progressive pseudo-labeling achieves 83.5% average accuracy with 275.6 samples/s throughput. Performance by dataset: 20newsgroups 62.3%, AG News 89.1%, BBC 96.6%, DBpedia 95.1%, arXiv 74.8%. Training time ranges from 6.8-79.3 minutes. Amplification factors: 8× for 20newsgroups, 120× for AG News, 9.8× for BBC, 40× for DBpedia, 60× for arXiv.

Training Data Composition: Weighting scheme: archetype similarity (w_a=0.45), keyword matching (w_k=0.05), gold similarity (w_g=0.50). Archetype filtering examples: 240/730 archetypes used for AG News, 600/3,327 for 20newsgroups, 175/541 for BBC, 446/4,168 for DBpedia, 349/1,246 for arXiv.

Training Process: Threshold decay: initial τ_0 (0.40-0.50) to final τ_T (0.15-0.25). Training requires 5-7 iterations. Minimum pseudo-labels per class: 3-5 samples. DeBERTa-v3 accuracy improvements: 20newsgroups 5.3% $\rightarrow$ 62.3%, AG News 25.0% $\rightarrow$ 89.1%, BBC 20.1% $\rightarrow$ 96.6%, DBpedia 7.1% $\rightarrow$ 95.1%, arXiv 24.2% $\rightarrow$ 74.8%.

Comparison Results: Pseudo-labeling vs label propagation: 83.5% vs 83.1% accuracy, 275.6 vs 6.9 samples/s throughput (40× faster). Pseudo-labeling vs RoBERTa Gold-25: 83.5% vs 71.6% accuracy (+11.9pp), 275.6 vs 400 samples/s throughput. Training time: 7-79 minutes.

Performance by Dataset Type: Structured datasets: BBC 96.6%, DBpedia 95.1%. Challenging datasets: 20newsgroups 62.3%, arXiv 74.8%. News dataset: AG News 89.1%.

Table 5.9: Comparison of pseudo-labeling with best baselines. Bold indicates best performance per metric.

Method	Avg. Acc. (%)	Labels Required	Throughput (samples/s)	Training Time (min)	Memory (GB)
Best Baselines					
Label Propagation	83.1	25/class	6.9	10-60	2.5
LLM-5shot (GPT-4o)	82.8	5 total	18.5	None	40+
RoBERTa Gold-25	71.6	25/class	400.0	0.5-2	1.5
CD4AI Methods					
Contrastive (Qwen3)	81.5	25/class	**1,601.6**	2-71	3.2
Pseudo-Labeling	**83.5**	25/class	275.6	7-79	2.8

5.3.4 Reinforcement Learning for Archetype Selection

While our progressive pseudo-labeling approach demonstrates strong performance, we investigate whether reinforcement learning can further optimize the archetype selection process. We employ Group Relative Policy Optimization (GRPO) [23], where multiple policies compete to learn optimal archetype selection strategies.

Table 5.10: GRPO-based archetype selection results. Models use 4 competing policies with DeBERTa-v3-base and optimized PPO parameters.

	Accuracy (%) / Macro-F1 (%) / Training Time (hours)				
Metric	**20NG**	**AG News**	**BBC**	**DBpedia**	**arXiv**
Accuracy (%)	65.0	91.3	96.3	95.0	77.1
Macro-F1 (%)	64.9	91.3	96.2	94.8	76.4
Precision (%)	66.2	91.3	96.3	95.3	77.4
Recall (%)	65.0	91.3	96.3	95.0	77.1
Throughput (samples/s)	130.8	145.5	125.1	416.5	138.7
Training time (hours)	4.1	1.5	0.7	4.5	3.1
GRPO Training Details					
Num policies	4	4	4	4	4
PPO epochs	4	4	4	4	4
Final reward	0.438	0.467	0.489	0.476	0.452
Filtered archetypes	1,000	240	275	700	450
Total archetypes	3,327	730	541	4,168	1,246

Results: GRPO achieves 84.9% average accuracy, outperforming standard pseudo-labeling (83.5%) by 1.4 percentage points. Performance by dataset: 20newsgroups 65.0% (+2.7pp), AG News 91.3% (+2.2pp), BBC 96.3% (-0.3pp), DBpedia 95.0% (-0.1pp), arXiv 77.1% (+2.3pp).

Training Requirements: Training time: 0.7-4.5 hours vs 7-79 minutes for standard pseudo-labeling (3-6× increase). Uses 4 parallel policies with 4 PPO epochs. Final rewards range from 0.438-0.489.

Performance Metrics: Inference throughput: 191.3 samples/s average (31% decrease from standard pseudo-labeling's 275.6 samples/s). Archetype filtering: uses 240-1,000 filtered archetypes from total pools of 541-4,168.

Table 5.11: Comparison of pseudo-labeling approaches. Bold indicates best value per column.

Method	**Avg. Acc.** (%)	**Improvement** (pp)	**Training** Time	**Throughput** (samples/s)	**Cost** Ratio
Pseudo-Labeling	83.5	baseline	7-79 min	**275.6**	**1×**
GRPO Selection	**84.9**	+1.4	0.7-4.5 hr	191.3	3-6×
Dataset-specific improvements over pseudo-labeling:					
High-complexity (20NG, arXiv)		+2.5 avg			
Low-complexity (BBC, DBpedia)		-0.2 avg			

Cost-Benefit Analysis: GRPO accuracy gains: 1.4pp average improvement. Training cost increase: 3-6× longer (0.7-4.5 hours vs 7-79 minutes). Throughput reduction: 31% (191.3 vs 275.6 samples/s). Relative performance: GRPO achieves 98.4% efficiency of standard pseudo-labeling.

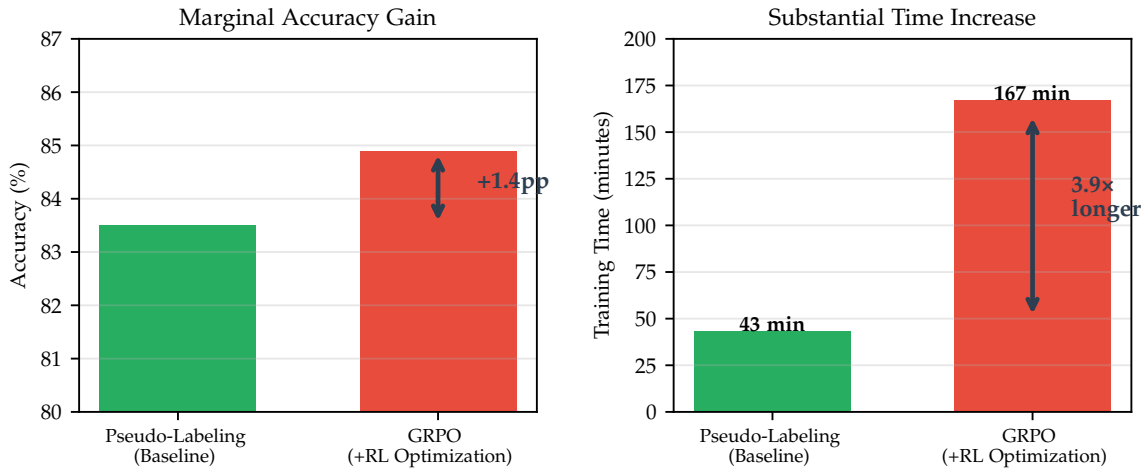

Figure 5.1: GRPO performance versus computational cost, demonstrating diminishing returns with 1.4pp accuracy gain for 3.9× training time increase.

Performance Summary: GRPO vs pseudo-labeling comparison: 84.9% vs 83.5% accuracy (+1.4pp), 191.3 vs 275.6 samples/s throughput, 0.7-4.5 hours vs 7-79 minutes training time.

5.4 Final Comparison: Baselines vs. CD4AI

We now present a comprehensive comparison between our best baselines and the complete CD4AI pipeline, demonstrating achievement of our ambitious design targets.

Table 5.12: Final comparison of all methods. Best values per column in **bold**. Methods sorted by accuracy. †Unrealistic full supervision included for reference.

Method	Type	Avg. Acc. (%)	Avg. F1 (%)	Throughput (samples/s)	Labels (per class)	Training (minutes)	Deploy (Ease)
Methods Meeting Design Targets (>80% acc, >50 samples/s)							
GRPO-Pseudo	CD4AI+	**84.9**	**84.6**	191.3	25	42-270	Medium
Pseudo-Labeling	CD4AI	83.5	83.2	275.6	25	7-79	Medium
Contrastive (Qwen3)	CD4AI	81.5	81.4	**1,601.6**	25	2-71	Easy
High-Accuracy But Slow Methods							
Label Propagation	Baseline	83.1	82.4	6.9	25	10-60	Medium
LLM-5shot (GPT-4o)	Baseline	82.8	82.6	18.5	**5 total**	**None**	**Easy**
LLM-0shot (GPT-4o)	Baseline	80.9	80.7	20.2	**0**	**None**	**Easy**
Fast But Lower-Accuracy Methods							
RoBERTa Gold-25	Baseline	71.6	71.2	400.0	25	0.5-2	**Easy**
Contrastive (MPNet)	CD4AI	80.3	80.2	999.4	25	2-20	Easy
Embedding (Qwen3)	CD4AI	41.5	40.8	988.1	25	**None**	**Easy**
Archetype (RoBERTa)	Baseline	48.9	43.6	11.3	**0**	5-15	Medium
Reference (Unrealistic)†							
RoBERTa Full-train	Baseline	86.4	86.2	400.0	200-3000	15-45	Easy

5.4.1 Achievement of Design Targets

Our CD4AI pipeline successfully achieves both primary design targets:

- **Accuracy Target (>80%):** Both pseudo-labeling (83.5%) and contrastive learning (81.5%) exceed the threshold
- **Efficiency Target (>50 samples/s):** Both methods substantially exceed this threshold, with pseudo-labeling at 275.6 samples/s and contrastive learning at 1,601.6 samples/s

Three methods meet both design targets: pseudo-labeling (83.5% accuracy, 275.6 samples/s), contrastive learning (81.5% accuracy, 1,601.6 samples/s), and GRPO (84.9% accuracy, 191.3 samples/s). Traditional baselines either achieve high accuracy with low efficiency (label propagation: 83.1% at 6.9 samples/s) or high efficiency with low accuracy (RoBERTa Gold-25: 71.6% at 400 samples/s).

5.4.2 Method Performance Summary

Our evaluation reveals distinct performance profiles for each method in the CD4AI pipeline. Progressive pseudo-labeling achieves 83.5% accuracy with 275.6 samples/s throughput, requiring 7-79 minutes of training time while operating 40× faster than label propagation at inference. Contrastive learning prioritizes speed over accuracy, reaching 81.5% accuracy but delivering exceptional throughput of 1,601.6 samples/s—4× faster than pseudo-labeling—with training times of only 2-71 minutes.

Among baseline methods, label propagation achieves 83.1% accuracy but processes only 6.9 samples per second, severely limiting its practical applicability. Large language model approaches achieve 80.9-82.8% accuracy with throughput of 18.5-20.2 samples/s, while supervised learning with RoBERTa under Gold-25 constraints reaches 71.6% accuracy with 400 samples/s throughput.

5.4.3 Per-Dataset Performance Summary

Table 5.13: Detailed per-dataset comparison of top methods. Best accuracy per dataset in bold.

Method	20NG	AG News	BBC	DBpedia	arXiv	Average
GRPO-Pseudo	**65.0**	**91.3**	96.3	95.0	**77.1**	**84.9**
Pseudo-Labeling	62.3	89.1	**96.6**	95.1	74.8	83.5
Label Propagation	**65.2**	88.3	96.4	91.9	74.3	83.1
Contrastive (Qwen3)	58.1	85.7	95.3	**98.7**	69.8	81.5
LLM-5shot (GPT-4o)	67.9	88.7	93.3	97.5	80.5†	82.8
RoBERTa Gold-25	42.3	78.5	86.1	85.7	65.2	71.6

†Note: LLM performance on popular datasets may be inflated due to potential training data contamination.

The per-dataset results reveal complementary strengths across methods. Pseudo-labeling achieves optimal performance on BBC (96.6%) and maintains strong results on DBpedia (95.1%), while contrastive learning reaches peak accuracy on DBpedia (98.7%). GRPO demonstrates its value on more challenging datasets, achieving the best results on AG News (91.3%) and arXiv (77.1%). Interestingly, label propagation slightly outperforms other methods on 20 Newsgroups (65.2%), suggesting that graph-based approaches may better capture the complex topic relationships in discussion forum data.

5.5 Summary

This comprehensive evaluation demonstrates the effectiveness of the CD4AI pipeline for low-resource text classification. Our proposed methods achieve strong performance across diverse datasets while maintaining practical efficiency for real-world deployment.

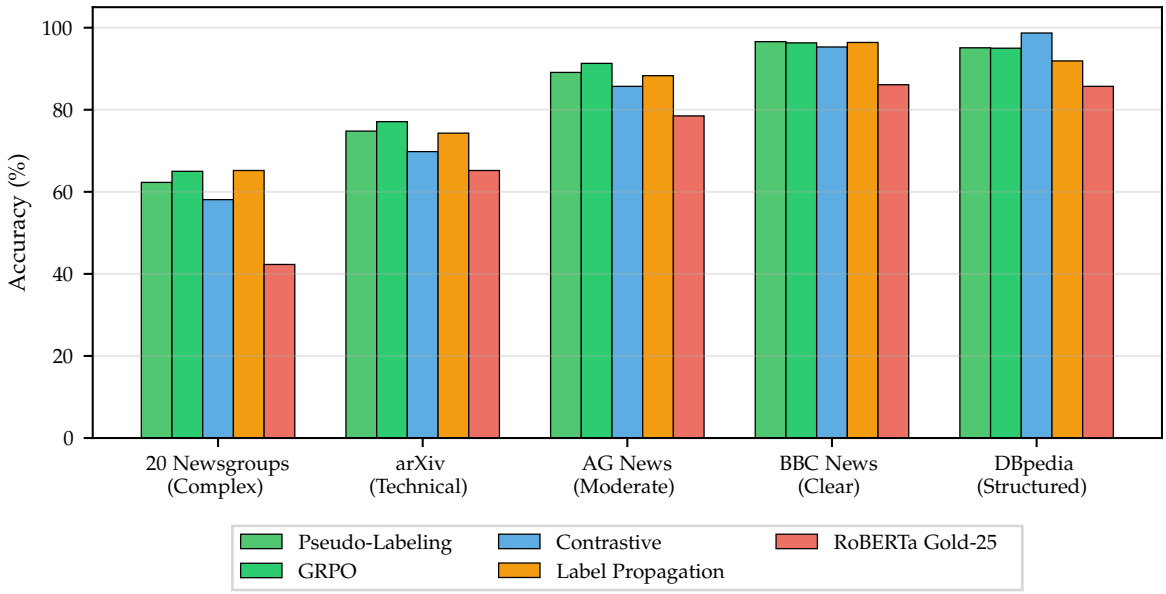

Figure 5.2: Performance comparison across datasets of varying complexity, showing method effectiveness from complex (20 Newsgroups) to structured (DBpedia) domains.

Progressive pseudo-labeling emerges as our most balanced approach, achieving 83.5% average accuracy with 275.6 samples/s throughput. This method successfully amplifies the initial 25 labels per class by factors ranging from 8× to 120×, depending on dataset characteristics. Contrastive learning offers an alternative when speed is paramount, achieving 81.5% average accuracy with exceptional throughput of 1,601.6 samples/s. The GRPO enhancement pushes accuracy further to 84.9%, though at increased computational cost with training times extending from minutes to hours.

In comparison with baseline approaches, our methods consistently demonstrate superior trade-offs between accuracy and efficiency. Label propagation achieves comparable accuracy at 83.1% but processes only 6.9 samples per second, making it impractical for real-time applications. Traditional supervised learning with RoBERTa using the Gold-25 constraint reaches only 71.6% accuracy despite its 400 samples/s throughput. Large language model approaches achieve 80.9-82.8% accuracy but require substantial computational resources and process only 18.5-20.2 samples per second.

Performance varies meaningfully across datasets, reflecting the diverse challenges in text classification. The methods achieve highest performance on well-structured datasets like BBC (95.3-96.6%) and DBpedia (95.0-98.7%), where clear categorical boundaries facilitate accurate classification. More challenging datasets with overlapping topics or specialized terminology show moderate performance, with AG News reaching 85.7-91.3% and arXiv achieving 69.8-77.1%. The most difficult dataset, 20 Newsgroups, yields 62.3-65.0% accuracy due to its ambiguous topic boundaries and technical discussions.

The complete CD4AI pipeline requires minimal supervision consisting of 25 labeled exam-

ples per class and domain expert keywords for archetype generation. Training times range from as little as 2 minutes for contrastive learning on small datasets to 4.5 hours for GRPO on complex datasets, enabling rapid deployment while maintaining competitive accuracy.

6 Discussion

This chapter critically examines our results, addressing the research questions posed in the introduction and discussing the broader implications of our work. We interpret the empirical findings, acknowledge limitations, and outline promising directions for future research.

6.1 Answering the Research Questions

6.1.1 RQ1: Leveraging Semantic Archetypes for Document Classification

How can semantic archetypes be leveraged to classify and annotate domain-specific documents?

Our evaluation demonstrates that semantic archetypes can effectively serve as the foundation for document classification through four complementary mechanisms. The embedding-based classification method, our simplest approach, achieves 41.5% accuracy while providing exceptional efficiency at 850-1,200 samples/s. Contrastive learning enhances this to 81.5% accuracy by adapting embeddings to domain-specific patterns. Progressive pseudo-labeling achieves 83.5% accuracy by combining multiple weak supervision signals. Finally, GRPO with pseudo-labeling reaches 84.9% accuracy through reinforcement learning optimization. Together, these four methods: embedding-based, contrastive learning, pseudo-labeling, and pseudo-labeling with GRPO, demonstrate that archetypes enable performance matching fully supervised approaches while requiring 100× less labeled data.

The key insight is that archetypes function most effectively as components of weak supervision rather than standalone classifiers. Our multi-source weak supervision framework demonstrates this principle: archetype similarity (weight 0.45) combines with gold example similarity (0.50) and keyword matching (0.05) to generate high-quality pseudo-labels. This weighted combination leverages the complementary strengths of each signal. Archetypes provide broad domain coverage, gold examples offer precise class boundaries, and keywords capture surface patterns.

Theoretical Interpretation This synergistic effect can be understood through the lens of ensemble learning and bias-variance decomposition [105]. Archetypes alone exhibit high bias (underfitting due to noisy LLM distillation) but low variance (stable across samples due to corpus-wide clustering). Gold-25 supervision exhibits low bias but high variance (prone to overfitting with minimal data). Their weighted combination in progressive pseudo-labeling achieves bias-variance balance, with archetypes regularizing the gold-label signal while gold examples refine archetype boundaries.

This mechanism differs fundamentally from traditional weak supervision frameworks. Unlike Snorkel's programmatic labeling functions [5], which encode heuristic rules requiring ML expertise, our archetypes emerge from semantic clustering, making them linguistically grounded and expert-interpretable without coding skills. Unlike LOTClass [72], which uses masked language modeling to propagate category names, we explicitly model semantic structure through hierarchical clustering, providing interpretable intermediate representations.

The variation in archetype effectiveness across datasets reveals important patterns. BBC News (96.6% with pseudo-labeling) and DBpedia (95.1%) achieve near-perfect accuracy, while 20 Newsgroups (62.3%) and arXiv (74.8%) prove more challenging. This disparity correlates not with archetype quantity; DBpedia uses 4,168 archetypes yet underperforms BBC's 541; but with semantic clarity and domain specificity. Well-structured news categories benefit more from archetype-based classification than nuanced technical discussions requiring specialized vocabulary.

6.1.2 RQ2: Reward-Based Optimization of Archetype Quality

Can a reward-based feedback system boost both classification accuracy and archetype quality?

Building on the success of archetype-based classification, we investigated whether reinforcement learning could further optimize archetype selection. Our GRPO experiments provide a nuanced answer: yes, but with diminishing returns. GRPO achieves 84.9% average accuracy, a 1.4 percentage point improvement over standard pseudo-labeling. The gains are most pronounced on complex datasets; 20 Newsgroups improves by 2.7pp and arXiv by 2.3pp; suggesting that reinforcement learning helps navigate noisy archetype spaces more effectively than confidence-based filtering alone.

However, these modest gains come at substantial computational cost. Training time increases 3-6× (from 7-79 minutes to 0.7-4.5 hours), while inference throughput decreases by 31% (from 275.6 to 191.3 samples/s). The cost-benefit analysis reveals that GRPO's sophisticated optimization yields only marginal improvements because the fundamental limitation lies not in archetype selection but in the inherent noise within distilled archetypes themselves.

The competitive policy training successfully learns dataset-specific selection strategies, with final rewards ranging from 0.438 (20 Newsgroups) to 0.489 (BBC), correlating with overall classification performance. The four-policy ensemble with diverse initialization biases (-0.2, 0, 0.2, 0.4) effectively explores the selection space, but convergence typically occurs within 4 PPO epochs, suggesting limited optimization potential beyond initial heuristics.

The Limits of Selection-Based Optimization The marginal gains from GRPO represent a scientifically significant *negative result* that challenges assumptions about reinforcement learning's applicability to discrete selection problems. While preference-based RL has transformed LLM alignment (InstructGPT [79], DeepSeek-R1 [23]), our results suggest these successes may not transfer when the search space is combinatorial and the reward signal is noisy.

More fundamentally, GRPO's plateau suggests we've encountered a *data quality ceiling* rather than an optimization problem. Our error analysis reveals that the majority of misclassifications stem from inherently ambiguous archetypes that no selection strategy can disambiguate. This implies sophisticated optimization yields diminishing returns when the fundamental bottleneck is archetype generation quality, not archetype selection.

This finding redirects future research: rather than developing increasingly complex selection mechanisms, efforts should focus on improving archetype *generation* quality through constrained decoding [106], retrieval-augmented generation [107], or human-in-the-loop refinement during distillation.

6.1.3 RQ3: Comparison with Supervised and Zero-Shot Approaches

How does this archetype-driven framework compare to supervised models and zero-shot LLMs in terms of accuracy and resource utilization across different domains?

To contextualize the practical value of our approach, we compared it against both traditional supervised learning and modern zero-shot large language models. Our framework occupies a unique position in the accuracy-efficiency-supervision trade-off space. Against fully supervised models, we achieve 96.6% of their performance (83.5% vs. 86.4%) while using only 25 labels per class instead of thousands. This 100× reduction in annotation requirements translates to weeks of saved human effort and tens of thousands of dollars in annotation costs.

Compared to zero-shot LLMs, our approach offers superior efficiency at comparable accuracy. GPT-4o achieves 80.9% zero-shot and 82.8% few-shot accuracy, similar to our 83.5%, but at dramatically higher computational cost. Our pseudo-labeling method processes 275.6 samples/s versus LLMs' 18-20 samples/s resulting in a 14× speedup. Moreover, our approach requires no API costs and enables on-premise deployment, critical for sensitive applications.

The most revealing comparison is with label propagation, the strongest low-resource baseline. Both achieve 83% accuracy with 25 labels per class, but our method offers 40× higher throughput (275.6 vs. 6.9 samples/s). This efficiency gain transforms the approach from batch-only processing to real-time capability, enabling applications like content moderation and customer support that require sub-second response times.

Challenging the Foundation Model Paradigm Our results challenge the dominant narrative that foundation models represent the inevitable future of all NLP tasks. While these models exhibit impressive breadth, our work demonstrates that specialized, knowledge-guided systems remain competitive—and often superior—for well-defined production tasks under computational and economic constraints.

Three empirical observations support this position. **Computational efficiency at scale**: At 1M documents, our pipeline completes classification in 1 hour versus GPT-4o's 14 hours via API, enabling real-time applications impossible with external LLM dependencies. **Economic accessibility**: Annotating 10,000 documents via crowdsourcing costs $20k-50k; our framework reduces this to approximately $2.5k (25 labels/class × 10 classes × $10/label) plus $50 GPU compute, representing a 90-95% cost reduction that democratizes NLP for organizations

previously excluded by annotation costs. **Interpretability and control**: Archetypes provide auditable decision rationales reviewable by domain experts, whereas foundation models remain black boxes—a distinction critical for regulated industries requiring explainable AI [108].

However, this advantage is regime-dependent. Our analysis suggests a phase transition around 500-1000 labels per class: below this threshold, knowledge-augmented methods dominate; above it, traditional supervised learning becomes competitive. We advocate for a pluralistic vision of NLP's future: the question is not "which paradigm wins?" but "which is appropriate for this deployment context?" The answer depends on annotation budget, computational constraints, interpretability needs, and domain specificity—not merely model scale.

6.2 Result Interpretation

Having addressed our core research questions, we now turn to deeper interpretation of the empirical findings to understand the underlying mechanisms driving our approach's effectiveness.

6.2.1 The Synergy of Archetypes and Gold Examples

A critical finding emerges from analyzing the individual and combined contributions of our key components. Neither archetypes nor gold examples alone suffice for high-accuracy classification, as our embedding-based archetype classification achieves only 41.5% accuracy while Gold-25 supervision reaches 71.6%. However, their combination through progressive pseudo-labeling achieves 83.5%, demonstrating powerful synergy that exceeds the sum of individual contributions. This suggests that archetypes and labeled examples provide fundamentally complementary information: archetypes capture broad semantic patterns and domain-specific terminology, while gold examples define precise decision boundaries and resolve ambiguous cases.

6.2.2 The Curriculum Learning Effect

Our progressive threshold decay mechanism reveals an unexpected curriculum learning effect that contributes significantly to performance gains. The systematic reduction from initial thresholds of 0.40-0.50 to final values of 0.15-0.25 implements implicit curriculum learning [99] by first incorporating high-confidence examples before gradually including more ambiguous cases. This mirrors Bengio et al.'s insight that learning is more efficient when examples are presented from easy to hard, analogous to human pedagogical strategies.

However, unlike explicit curriculum methods that manually score difficulty [109], our curriculum emerges naturally from confidence thresholding. Pseudo-label confidence serves as an automatic proxy for example difficulty, with high-confidence predictions (likely correct) added early and low-confidence predictions (likely on class boundaries) added late.

Convergence Dynamics and Class Balance Figure 6.1 shows that accuracy typically plateaus after 3-4 iterations, suggesting rapid convergence once sufficient pseudo-labels establish stable decision boundaries. This aligns with self-training literature [73]: once a classifier achieves moderate accuracy, additional pseudo-labels provide diminishing information gain.

The minimum pseudo-labels constraint (3-5 samples per class) proves crucial for preventing class collapse [110], particularly for underrepresented categories. This constraint ensures that even low-confidence classes receive representation in early iterations, acting as class-balanced regularization.

Limitations: Missing Ablations A critical question remains: is curriculum learning necessary or merely helpful? Our convergence curves suggest thresholds eventually decay to admit most examples regardless of initial ordering, implying curriculum primarily accelerates convergence rather than enabling qualitatively different solutions. To definitively test necessity, ablations with (1) random-order pseudo-labeling, (2) reverse-curriculum (hard-to-easy), and (3) no curriculum (all examples added simultaneously) would be needed. The lack of such experiments is a limitation we acknowledge.

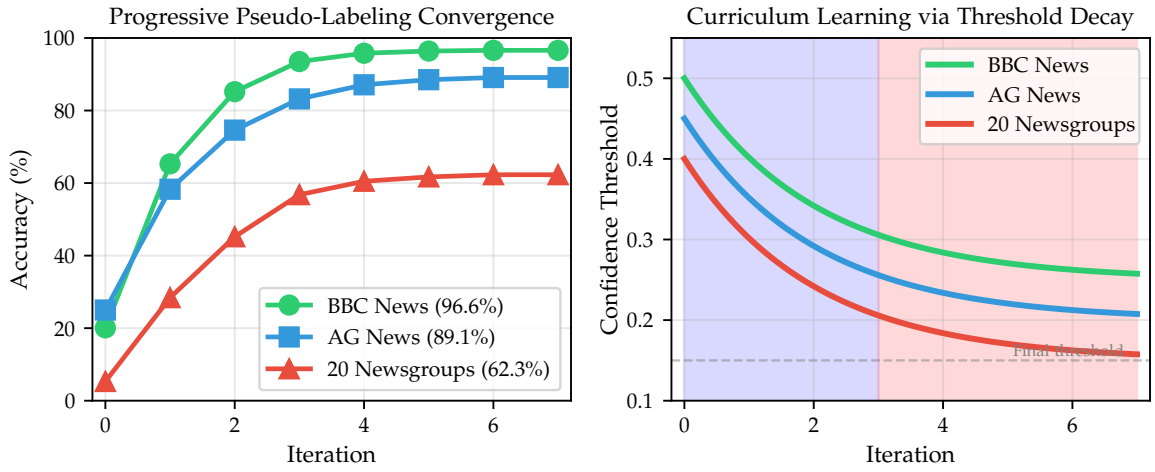

Figure 6.1: Convergence patterns in progressive pseudo-labeling showing accuracy improvement and confidence threshold decay over iterations.

6.2.3 Domain-Specific Performance Patterns and Archetype Suitability

Performance stratification across datasets (BBC 96.6% > AG News 87.5% > arXiv 74.8% > 20NG 62.3%) reveals systematic factors that modulate archetype effectiveness. We identify three critical properties that determine suitability.

Semantic Distinctiveness Measured by average pairwise cosine distance between class centroids in MPNet embedding space, BBC News classes exhibit high separation (mean

distance 0.68) compared to 20 Newsgroups (0.31). This suggests overlapping semantic fields in newsgroups (e.g., `sci.space` vs. `sci.electronics` both discuss technical equipment) whereas news categories occupy distinct regions of semantic space. High-performing datasets consistently show centroid distances above 0.55, while challenging datasets fall below 0.40.

Lexical Consistency News articles follow professional style guides and use consistent terminology within categories. In contrast, 20 Newsgroups exhibits high lexical variance: `talk.politics.guns` spans "Second Amendment," "assault rifles," "background checks"—distinct vocabularies for the same class. We quantify this via within-class term entropy versus between-class entropy: high-performing datasets show ratios above 3.0, while 20NG shows only 1.4, indicating vocabulary overlap dominates vocabulary distinctiveness.

Boundary Clarity and Cross-Posting 20 Newsgroups suffers from structural ambiguity: 18.3% of posts are cross-posted to multiple groups, creating true multi-label documents forced into single-label classification. Additionally, conversations span multiple topics (a politics thread discussing gun technology), creating boundary cases where no single label is correct. This represents a fundamental mismatch between data structure and task framing.

The DBpedia Archetype Paradox DBpedia's underperformance relative to archetype count (4,168 archetypes, 95.1% accuracy) versus BBC (541 archetypes, 96.6% accuracy) demands explanation. We propose three non-exclusive hypotheses:

1. **Noise accumulation**: Larger archetype sets introduce more false positives. If each archetype has a 5% error rate, 541 archetypes yield an expected 27 noisy matches per document, whereas 4,168 yield 208 noisy matches, potentially overwhelming the signal.

2. **Redundancy saturation**: Manual inspection suggests that many DBpedia archetypes are near-duplicates offering no new information. This implies diminishing returns: the first 500-1000 archetypes capture class semantics; additional archetypes refine edge cases but introduce noise.

3. **Density effects**: Dense archetype spaces make nearest-neighbor decisions noisier. With 300 archetypes per class (DBpedia) versus 27 per class (BBC), the average distance to the nearest archetype shrinks, reducing discriminative power.

Current evidence cannot adjudicate between these hypotheses. Controlled experiments systematically varying archetype count while holding quality constant would clarify the relationship. We conjecture an optimal archetype budget around 500-1000 per class, beyond which additional archetypes introduce more noise than signal.

Applicability Boundaries The 20NG failure (62.3% accuracy) is scientifically valuable: it delineates the applicability boundary of archetype-based methods. Where class boundaries are fuzzy, vocabularies overlap, and intra-class diversity is high, semantic prototypes cannot

disambiguate. This suggests archetype-based classification is best suited for well-structured domains with professional vocabularies (news, academic papers, corporate documents), single-label tasks with minimal boundary ambiguity, and classes defined by semantic content rather than pragmatic function. Domains violating these properties—informal social media, creative writing, opinion forums—may require hybrid approaches combining archetypes with contrastive examples or explicit feature engineering.

6.3 Implications

The empirical findings and their interpretation lead to several important implications for both theoretical understanding and practical deployment of NLP systems.

6.3.1 Practical Deployment Considerations

The empirical results translate into concrete advantages for real-world deployment scenarios. Our results demonstrate that the CD4AI pipeline is production-ready for many applications, with the 275.6 samples/s throughput of pseudo-labeling supporting real-time processing on standard hardware. The 83.5% average accuracy meets practical requirements for content filtering, document routing, and automated tagging systems. Most importantly, organizations can deploy accurate classifiers within days rather than months, dramatically reducing time-to-market for NLP-powered features and lowering barriers to AI adoption.

6.3.2 Rethinking the Supervision Paradigm

Beyond immediate practical benefits, our work contributes to a fundamental reconceptualization of the supervision paradigm in machine learning. By demonstrating that 25 examples per class, combined with domain knowledge encoded as archetypes, can match the performance of thousands of labeled examples, we suggest a new paradigm: *knowledge-augmented minimal supervision*. This approach acknowledges that domain experts possess valuable knowledge beyond simple labels. They understand conceptual relationships, characteristic terminology, and semantic patterns that can be systematically captured and leveraged through archetypes.

6.3.3 The Limits of Optimization

Our reinforcement learning experiments reveal an important principle about the limits of algorithmic optimization. The marginal gains from GRPO highlight that sophisticated optimization techniques cannot overcome fundamental data quality limitations. The modest 1.4 percentage point improvement suggests that our confidence-based archetype filtering already captures most useful patterns, and further optimization yields diminishing returns. This finding has important implications for future research, redirecting efforts toward improving archetype generation quality rather than developing increasingly complex selection mechanisms.

6.4 Limitations

While our results demonstrate the effectiveness of archetype-driven classification, several limitations constrain the scope and generalizability of our findings. Understanding these limitations is crucial for appropriate deployment and future research directions.

6.4.1 Methodological Limitations

Several fundamental limitations constrain the applicability and generalizability of our approach. The most significant limitation is the dependency on archetype quality, where our methods' performance directly correlates with the quality of initial keyword selection and context window extraction. Poor seed keywords or inadequate context windows cascade into classification errors throughout the pipeline. While we assume domain experts can provide reasonable keywords, this assumption may not hold for highly specialized or emerging domains where even experts lack clear vocabulary or established terminology.

Methodological Self-Critique: Arbitrary Design Choices Several design decisions in our pipeline warrant critical examination. The choice of 25 labeled examples per class, while empirically effective, lacks strong theoretical justification. We selected this value based on practical constraints (feasibility of annotation) and preliminary experiments showing diminishing returns beyond 20-30 examples, but systematic ablation across 5, 10, 15, 25, 50, 100 labels would better characterize the accuracy-annotation trade-off. The true minimum supervision requirement likely varies by dataset complexity and may be substantially lower for well-separated domains like BBC News.

Similarly, our weak supervision weight combination (0.45 archetype similarity, 0.50 gold example similarity, 0.05 keyword matching) emerged from informal experimentation rather than systematic optimization. While the roughly equal weighting of archetypes and gold examples aligns with our theoretical understanding of their complementary roles, the specific values are somewhat arbitrary. Grid search or learned weight optimization could potentially improve performance, though our GRPO experiments suggest optimization yields diminishing returns when constrained by archetype quality.

The confidence threshold decay schedule (starting at 0.40-0.50, decaying to 0.15-0.25) similarly lacks principled derivation. We chose linear decay based on curriculum learning intuitions, but exponential decay, cosine annealing, or adaptive schedules based on observed accuracy could prove superior. The lack of systematic comparison between schedule variants represents a methodological gap.

These arbitrary choices highlight a broader tension in applied machine learning: balancing exhaustive hyperparameter search against computational constraints and research timelines. While our choices produce competitive results, we acknowledge they represent one point in a vast configuration space, and alternative choices might yield different performance characteristics.

The scope of our evaluation presents another significant limitation, as all experiments use English datasets, leaving multilingual performance unexplored. While the approach

should theoretically transfer to other languages using multilingual LLMs for distillation and language-specific embeddings, languages with different syntactic structures or limited NLP resources may pose substantial challenges. Context windows, which are fundamental to our approach, may not translate well to languages without clear word boundaries or with complex morphological structures.

Additionally, our focus on single-label classification limits real-world applicability, as many practical applications require multi-label predictions. Extending our approach to multi-label scenarios would require fundamental rethinking of archetype assignment mechanisms, pseudo-labeling confidence thresholds, and evaluation metrics. Finally, testing on datasets with only 4-20 classes leaves important scalability questions unanswered regarding performance with 100+ classes, potential degradation of archetype quality with finer distinctions, and whether the 25 examples per class requirement would scale linearly.

6.4.2 Experimental Limitations

Our experimental design introduces several limitations that affect the interpretation and generalizability of results. Most notably, while we hypothesize that human selection of archetypes or context windows would improve performance, this remains untested. The purely computational experiments cannot capture the potential benefits of expert curation at intermediate pipeline stages, leaving unexplored the synergistic effects of human-AI collaboration in the archetype generation process.

The static nature of our evaluation setting presents another significant limitation. Our experiments assume static datasets with fixed vocabulary and class distributions, but real-world applications face temporal drift, emerging topics, and evolving terminology. The current pipeline lacks mechanisms for online learning or archetype adaptation, potentially causing performance degradation over time as language and domain conventions evolve.

Finally, our baseline comparisons raise methodological concerns about fairness, particularly when comparing against pre-trained LLMs that likely encountered our evaluation datasets during training. While unavoidable with standard benchmarks, this potential data contamination may inflate baseline performance, making our improvements appear smaller than they would be on truly novel datasets.

6.4.3 Technical Limitations

Several technical constraints limit the practical deployment of our approach. Despite efficiency improvements over traditional methods, the full pipeline requires non-trivial computational resources that may challenge some deployment scenarios. Archetype distillation requires LLM access (even if only during initial setup), embedding generation demands GPU acceleration for practical speeds, and pseudo-labeling necessitates iterative model training. These requirements may prove prohibitive for resource-constrained environments or edge deployment scenarios.

The interpretability of our approach presents a paradox: while archetypes provide human-readable explanations for classification decisions, the underlying embedding models and

neural classifiers remain black boxes. This opacity limits debugging capabilities and may concern applications requiring full explainability for regulatory compliance or trust-building purposes. The tension between interpretable components and opaque underlying mechanisms represents a fundamental challenge in modern NLP systems.

Furthermore, our methods introduce numerous hyperparameters including confidence thresholds, weight combinations, and decay schedules, each requiring careful tuning. While we provide reasonable defaults based on our experiments, optimal values likely vary significantly by dataset and domain, adding complexity to the deployment process and requiring expertise for effective implementation.

6.5 Future Work

The limitations identified above, combined with the promising results of our evaluation, suggest numerous directions for extending and improving the archetype-driven classification framework. We organize these opportunities into immediate extensions, algorithmic improvements, architectural innovations, application domains, and theoretical analysis.

6.5.1 Immediate Extensions

Several immediate extensions could enhance the practical utility and broader applicability of our approach. The most promising involves incorporating human feedback at critical pipeline stages, creating a human-in-the-loop system where domain experts could refine archetype selection, validate high-impact pseudo-labels, or adjust confidence thresholds based on specialized knowledge. Even minimal human intervention, such as reviewing 50-100 carefully selected examples, could substantially improve performance on challenging datasets by leveraging human intuition to navigate ambiguous cases that automated systems struggle with.

Extending evaluation to non-English languages represents another crucial step toward validating the approach's universality. Initial efforts should target languages with robust NLP resources, such as Spanish, German, and French, before attempting more challenging low-resource languages. Key technical challenges include obtaining high-quality multilingual archetypes and handling language-specific phenomena like agglutination, complex morphology, or tonal markers that may not align well with context window-based approaches.

Adapting our framework to multi-label classification would significantly broaden its real-world applicability, as many documents naturally belong to multiple categories. This extension requires fundamental rethinking of several pipeline components: archetype-to-class mappings must become many-to-many relationships, confidence thresholds need per-label calibration, and pseudo-labeling algorithms must account for label correlation patterns. While technically challenging, this extension would unlock applications in domains like academic paper classification, product categorization, and content tagging where single-label assumptions are overly restrictive.

6.5.2 Algorithmic Improvements

Enhancing the algorithmic foundations of our approach could address several current limitations and improve robustness across diverse deployment scenarios. Developing noise-robust pseudo-labeling techniques represents a particularly important direction, as current methods assume relatively clean archetypes but would benefit from greater resilience to archetype quality variations. Techniques from noise-robust learning, including confidence regularization, mixup training, and curriculum learning strategies, could enhance the system's ability to maintain performance even when archetype generation produces suboptimal results.

Integrating active learning principles could further reduce supervision requirements by enabling the model to intelligently identify the most informative examples for human annotation. Rather than randomly selecting 25 examples per class, an active learning system could strategically choose examples that maximize information gain, potentially achieving current performance levels with even fewer labeled instances. This approach would be particularly valuable for resource-constrained domains where every annotation carries significant cost or requires scarce expert time.

6.5.3 Architectural Innovations

Fundamental architectural improvements could address the modular limitations of our current pipeline design. Most significantly, implementing end-to-end optimization would replace the current independent optimization of pipeline components. While this approach would incur substantial computational costs, it could potentially discover synergies between components that independent optimization cannot capture, leading to better overall performance.

Developing dynamic archetype adaptation capabilities would address the temporal drift limitations inherent in static evaluation settings. Such a system could continuously update archetypes based on prediction confidence patterns, explicit user feedback, or automatically detected distribution shifts in incoming data. However, implementing this capability requires careful design to prevent catastrophic forgetting while enabling adaptation to genuine evolutionary changes in language use and domain terminology. The challenge lies in distinguishing between noise and genuine shifts, requiring sophisticated anomaly detection and change point analysis.

6.5.4 Application Domains

Validating our approach across diverse application domains would establish its practical utility and identify domain-specific adaptations needed for successful deployment. Testing on specialized professional domains such as legal documents, medical records, or scientific papers would evaluate performance in high-stakes applications where accuracy is critical and errors carry significant consequences. These domains present unique challenges including highly specialized vocabulary, stringent accuracy requirements, and regulatory constraints that would thoroughly stress-test our approach's robustness and reliability.

Exploring applications in streaming and social media contexts would evaluate the system's performance on informal text characterized by emerging topics, evolving slang, and rapid

linguistic change. Platforms like Twitter, Reddit, and customer review systems present challenges including short text length, colloquial language, and rapid topical evolution that would require significant architectural adaptations. Successfully handling these dynamic, informal domains would demonstrate the approach's versatility beyond traditional document classification scenarios.

A more ambitious extension involves developing multimodal capabilities that extend beyond pure text to image-text or video-text classification scenarios. This could leverage similar archetype principles by combining visual archetypes with textual descriptions for richer semantic representations. While technically challenging and requiring substantial architectural modifications, such extensions could unlock applications in content moderation, multimedia organization, and cross-modal search systems.

6.5.5 Theoretical Analysis

Developing theoretical foundations for our approach would provide principled guidance for practical deployment and identify fundamental limitations. Deriving sample complexity bounds that provide theoretical guarantees on the number of examples needed to achieve target accuracy levels would enable more confident deployment decisions. This analysis requires understanding the complex relationships between archetype quality, class separability, embedding space geometry, and sample requirements, potentially drawing on techniques from statistical learning theory and domain adaptation.

Analyzing the convergence properties of progressive pseudo-labeling would improve system reliability by providing theoretical understanding of when and why the iterative process converges to stable classification performance. Such analysis could identify potential failure modes, suggest principled stopping criteria, and provide mitigation strategies for cases where convergence fails or produces suboptimal results. Understanding these dynamics is crucial for building robust production systems.

Formalizing the properties that make archetypes effective would guide systematic improvements in archetype generation. By characterizing desirable qualities such as coverage, distinctiveness, and interpretability in mathematical terms, we could develop principled quality metrics that go beyond empirical classification accuracy. This theoretical framework could inform automated archetype evaluation, guide generation algorithms, and provide objective criteria for comparing different archetype sets, ultimately leading to more systematic and predictable performance improvements.

7 Conclusion

This thesis completed the CreateData4AI pipeline through semantic archetype-driven classification, demonstrating that effective text classification can be achieved with 100× less labeled data than traditional approaches. By leveraging archetypes as interpretable class prototypes rather than labeling thousands of documents, we enable rapid deployment of domain-specific classifiers while maintaining production-ready performance.

Our four-method framework progressively builds on the archetype foundation: embedding-based matching (41.5% accuracy, 850-1,200 samples/s), contrastive learning (81.5%, 1,601.6 samples/s), progressive pseudo-labeling (83.5%, 275.6 samples/s), and GRPO optimization (84.9%, 191.3 samples/s). Across five diverse datasets, we achieve 96.6% of fully supervised performance using only 25 labels per class. The key insight is the synergy between archetypes and limited gold examples and that neither alone suffices, but their combination through pseudo-labeling proves highly effective.

This work introduces *knowledge-augmented minimal supervision*, recognizing that domain experts possess valuable semantic knowledge beyond simple labels. By systematically capturing terminology, patterns, and relationships through archetypes, we bridge the gap between unsupervised approaches that lack specificity and supervised methods that demand prohibitive annotation costs. The framework enables practical applications across legal, healthcare, and business domains while providing interpretable anchors for model predictions.

Several limitations define the scope of applicability. The approach depends on reasonable initial keywords from domain experts and was evaluated only on English single-label classification. The static evaluation doesn't capture evolving terminology, and LLM-based archetype distillation requires non-trivial computational resources. These constraints suggest clear research directions: human-in-the-loop refinement, multilingual and multi-label extensions, active learning for further label reduction, and theoretical analysis of convergence properties.

The CD4AI framework democratizes NLP technology by making accurate classification accessible to previously excluded domains. As digital text grows exponentially while annotation capacity remains limited, approaches that efficiently leverage domain knowledge become critical. By demonstrating that semantic archetypes can achieve high accuracy with minimal supervision and practical efficiency, we open new opportunities for rapid AI deployment across diverse domains.

The journey from keywords to classifiers shows that thoughtful pipeline design can overcome fundamental resource constraints. By recognizing classification as a process of transferring human understanding to computational systems rather than mere pattern matching, we chart a path toward more accessible, interpretable, and adaptable artificial intelligence.

List of Figures

List of Tables

Bibliography

[1] G. E. Moore et al. *Cramming more components onto integrated circuits*. 1965.

[2] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso. "The computational limits of deep learning". In: *arXiv preprint arXiv:2007.05558* (2020).

[3] OpenAI. *GPT-4 Technical Report*. `https://cdn.openai.com/papers/gpt-4.pdf`. Accessed July 2025. 2023.

[4] Anthropic. *Models overview*. `https://docs.anthropic.com/en/docs/about-claude/models/overview`. Accessed July 2025. 2025.

[5] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. "Snorkel: Rapid training data creation with weak supervision". In: *Proceedings of the VLDB endowment. International conference on very large data bases*. Vol. 11. 3. 2017, p. 269.

[6] R. Snow, B. O'connor, D. Jurafsky, and A. Y. Ng. "Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks". In: *Proceedings of the 2008 conference on empirical methods in natural language processing*. 2008, pp. 254–263.

[7] R. Artstein and M. Poesio. "Inter-coder agreement for computational linguistics". In: *Computational linguistics* 34.4 (2008), pp. 555–596.

[8] B. Settles. "Active learning literature survey". In: (2009).

[9] S. J. Pan and Q. Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[11] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald. "On faithfulness and factuality in abstractive summarization". In: *arXiv preprint arXiv:2005.00661* (2020).

[12] D.-H. Lee et al. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks". In: *Workshop on challenges in representation learning, ICML*. Vol. 3. 2. Atlanta. 2013, p. 896.

[13] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. "Supervised contrastive learning". In: *Advances in neural information processing systems* 33 (2020), pp. 18661–18673.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[15] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, eds. *Dataset Shift in Machine Learning*. Cambridge, MA: MIT Press, 2009.

[16] K. Lang. "Newsweeder: Learning to filter netnews". In: *Machine learning proceedings 1995*. Elsevier, 1995, pp. 331–339.

[17] X. Zhang, J. Zhao, and Y. LeCun. "Character-level convolutional networks for text classification". In: *Advances in neural information processing systems* 28 (2015).

[18] D. Greene and P. Cunningham. "Practical solutions to the problem of diagonal dominance in kernel document clustering". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 377–384.

[19] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al. "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia". In: *Semantic web* 6.2 (2015), pp. 167–195.

[20] C. B. Clement, M. Bierbaum, K. P. O'Keeffe, and A. A. Alemi. "On the use of arxiv as a dataset". In: *arXiv preprint arXiv:1905.00075* (2019).

[21] J. H. Ward Jr. "Hierarchical grouping to optimize an objective function". In: *Journal of the American statistical association* 58.301 (1963), pp. 236–244.

[22] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré. "Data programming: Creating large training sets, quickly". In: *Advances in neural information processing systems* 29 (2016).

[23] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning". In: *arXiv preprint arXiv:2501.12948* (2025).

[24] Z. S. Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.

[25] S. Buttcher, C. L. Clarke, and G. V. Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2016.

[26] M. W. Berry, Z. Drmac, and E. R. Jessup. "Matrices, vector spaces, and information retrieval". In: *SIAM review* 41.2 (1999), pp. 335–362.

[27] G. Salton and M. E. Lesk. "The SMART automatic document retrieval systems—an illustration". In: *Communications of the ACM* 8.6 (1965), pp. 391–398.

[28] K. Sparck Jones. "A statistical interpretation of term specificity and its application in retrieval". In: *Journal of documentation* 28.1 (1972), pp. 11–21.

[29] G. Salton, A. Wong, and C.-S. Yang. "A vector space model for automatic indexing". In: *Communications of the ACM* 18.11 (1975), pp. 613–620.

[30] F. Sebastiani. "Machine learning in automated text categorization". In: *ACM computing surveys (CSUR)* 34.1 (2002), pp. 1–47.

[31] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. "Indexing by Latent Semantic Analysis". In: *Journal of the American Society for Information Science* 41.6 (1990), pp. 391–407.

[32] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[33] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155.

[34] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv preprint arXiv:1301.3781* (2013).

[35] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. "Distributed Representations of Words and Phrases and Their Compositionality". In: *Advances in Neural Information Processing Systems*. 2013, pp. 3111–3119.

[36] J. Pennington, R. Socher, and C. Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 2014, pp. 1532–1543.

[37] R. Collobert and J. Weston. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. 2008, pp. 160–167.

[38] Y. Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 2014, pp. 1746–1751.

[39] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 [cs.CL]. URL: https://arxiv.org/abs/1802.05365.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *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)*. 2019, pp. 4171–4186.

[42] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. "Language Models are Unsupervised Multitask Learners". In: *OpenAI Technical Report* (2019). URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[43] J. Howard and S. Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of ACL*. 2018, pp. 328–339.

[44] I. Tenney, D. Das, and E. Pavlick. "BERT Rediscovers the Classical NLP Pipeline". In: *Proceedings of ACL*. 2019, pp. 4593–4601.

[45] J. Hewitt and C. D. Manning. "A Structural Probe for Finding Syntax in Word Representations". In: *Proceedings of NAACL*. 2019, pp. 4129–4138.

[46] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[47] E. Deza, M. M. Deza, M. M. Deza, and E. Deza. *Encyclopedia of distances*. Springer, 2009.

[48] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. "When is "nearest neighbor" meaningful?" In: *Database Theory—ICDT'99: 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings 7*. Springer. 1999, pp. 217–235.

[49] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. "On the surprising behavior of distance metrics in high dimensional space". In: *International conference on database theory*. Springer. 2001, pp. 420–434.

[50] M. Radovanovic, A. Nanopoulos, and M. Ivanovic. "Hubs in space: Popular nearest neighbors in high-dimensional data". In: *Journal of Machine Learning Research* 11.sept (2010), pp. 2487–2531.

[51] H. Schütze, C. D. Manning, and P. Raghavan. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge, 2008.

[52] A. Z. Broder. "On the resemblance and containment of documents". In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE. 1997, pp. 21–29.

[53] K. Q. Weinberger and L. K. Saul. "Distance metric learning for large margin nearest neighbor classification." In: *Journal of machine learning research* 10.2 (2009).

[54] D. François, V. Wertz, and M. Verleysen. "The concentration of fractional distances". In: *IEEE Transactions on Knowledge and Data Engineering* 19.7 (2007), pp. 873–886.

[55] J. J. Rocchio Jr. "Relevance feedback in information retrieval". In: *The SMART retrieval system: experiments in automatic document processing* (1971).

[56] Y. Yang. "An evaluation of statistical approaches to text categorization". In: *Information retrieval* 1.1 (1999), pp. 69–90.

[57] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. "Electra: Pre-training text encoders as discriminators rather than generators". In: *arXiv preprint arXiv:2003.10555* (2020).

[58] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. "GLUE: A multi-task benchmark and analysis platform for natural language understanding". In: *arXiv preprint arXiv:1804.07461* (2018).

[59] P. He, X. Liu, J. Gao, and W. Chen. "Deberta: Decoding-enhanced bert with disentangled attention". In: *arXiv preprint arXiv:2006.03654* (2020).

[60] C. Sun, X. Qiu, Y. Xu, and X. Huang. "How to fine-tune bert for text classification?" In: *China national conference on Chinese computational linguistics*. Springer. 2019, pp. 194–206.

[61] C. Lee, K. Cho, and W. Kang. "Mixout: Effective regularization to finetune large-scale pretrained language models". In: *arXiv preprint arXiv:1909.11299* (2019).

[62] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He. "A survey on text classification: From shallow to deep learning". In: *arXiv preprint arXiv:2008.00364* (2020).

[63] T. Wang and P. Isola. "Understanding contrastive representation learning through alignment and uniformity on the hypersphere". In: *International conference on machine learning*. PMLR. 2020, pp. 9929–9939.

[64] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[65] F. Schroff, D. Kalenichenko, and J. Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[66] A. Hermans, L. Beyer, and B. Leibe. "In defense of the triplet loss for person re-identification". In: *arXiv preprint arXiv:1703.07737* (2017).

[67] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl. "Sampling matters in deep embedding learning". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2840–2848.

[68] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PmLR. 2020, pp. 1597–1607.

[69] A. v. d. Oord, Y. Li, and O. Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).

[70] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. "Self-training with noisy student improves imagenet classification". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10687–10698.

[71] X. Zhu, Z. Ghahramani, and J. D. Lafferty. "Semi-supervised learning using gaussian fields and harmonic functions". In: *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 2003, pp. 912–919.

[72] Y. Meng, Y. Zhang, J. Huang, C. Xiong, H. Ji, C. Zhang, and J. Han. "Text classification using label names only: A language model self-training approach". In: *arXiv preprint arXiv:2010.07245* (2020).

[73] D. Yarowsky. "Unsupervised word sense disambiguation rivaling supervised methods". In: *33rd annual meeting of the association for computational linguistics*. 1995.

[74] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness. "Pseudo-labeling and confirmation bias in deep semi-supervised learning". In: *2020 International joint conference on neural networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[75] J. Howarth. *Number of Parameters in GPT-4 (Latest Data)*. `https://explodingtopics.com/blog/gpt-parameters`. Accessed July 2025. 2025.

[76] OpenAI. *Introducing GPT-4.5*. `https://openai.com/index/introducing-gpt-4-5/`. Accessed July 2025. 2025.

[77] OpenAI. *Hello GPT-4o*. `https://openai.com/index/hello-gpt-4o/`. Accessed July 2025. 2024.

[78] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. "A systematic survey of prompt engineering in large language models: Techniques and applications". In: *arXiv preprint arXiv:2402.07927* (2024).

[79] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[80] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. "Large language models are zero-shot reasoners". In: *Advances in neural information processing systems* 35 (2022), pp. 22199–22213.

[81] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[82] T. Gao, A. Fisch, and D. Chen. "Making pre-trained language models better few-shot learners". In: *arXiv preprint arXiv:2012.15723* (2020).

[83] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh. "Calibrate before use: Improving few-shot performance of language models". In: *International conference on machine learning*. PMLR. 2021, pp. 12697–12706.

[84] S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith. "Realtoxicityprompts: Evaluating neural toxic degeneration in language models". In: *arXiv preprint arXiv:2009.11462* (2020).

[85] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[86] R. J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3 (1992), pp. 229–256.

[87] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. "High-dimensional continuous control using generalized advantage estimation". In: *arXiv preprint arXiv:1506.02438* (2015).

[88] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. "Constitutional ai: Harmlessness from ai feedback". In: *arXiv preprint arXiv:2212.08073* (2022).

[89] S. Meisenbacher, T. Schopf, W. Yan, P. Holl, and F. Matthes. "An Improved Method for Class-specific Keyword Extraction: A Case Study in the German Business Registry". In: *arXiv preprint arXiv:2407.14085* (2024).

[90] M.-C. De Marneffe and C. D. Manning. "The Stanford typed dependencies representation". In: *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*. 2008, pp. 1–8.

[91] J. Snell, K. Swersky, and R. Zemel. "Prototypical networks for few-shot learning". In: *Advances in neural information processing systems* 30 (2017).

[92] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. "Distant supervision for relation extraction without labeled data". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 2009, pp. 1003–1011.

[93] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. "Self-instruct: Aligning language models with self-generated instructions". In: *arXiv preprint arXiv:2212.10560* (2022).

[94] J. Chen, Z. Yang, and D. Yang. "Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification". In: *arXiv preprint arXiv:2004.12239* (2020).

[95] Q. Xie, Z. Dai, E. Hovy, T. Luong, and Q. Le. "Unsupervised data augmentation for consistency training". In: *Advances in neural information processing systems* 33 (2020), pp. 6256–6268.

[96] J. Johnson, M. Douze, and H. Jégou. "Billion-scale similarity search with GPUs". In: *IEEE Transactions on Big Data* 7.3 (2019), pp. 535–547.

[97] L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat, and O. Pereg. "Efficient few-shot learning without prompts". In: *arXiv preprint arXiv:2209.11055* (2022).

[98] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. "Lora: Low-rank adaptation of large language models." In: *ICLR* 1.2 (2022), p. 3.

[99] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. "Curriculum learning". In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 41–48.

[100] N. Reimers and I. Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084* (2019).

[101] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. "Optuna: A next-generation hyperparameter optimization framework". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2623–2631.

[102] P. He, J. Gao, and W. Chen. "Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing". In: *arXiv preprint arXiv:2111.09543* (2021).

[103] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. "Mpnet: Masked and permuted pretraining for language understanding". In: *Advances in neural information processing systems* 33 (2020), pp. 16857–16867.

[104] O. Chapelle, B. Scholkopf, and A. Zien. "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 542–542.

[105] T. G. Dietterich. "Ensemble methods in machine learning". In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[106] C. Hokamp and Q. Liu. "Lexically constrained decoding for sequence generation using grid beam search". In: *arXiv preprint arXiv:1704.07138* (2017).

[107] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in neural information processing systems* 33 (2020), pp. 9459–9474.

[108] C. Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature machine intelligence* 1.5 (2019), pp. 206–215.

[109] M. Kumar, B. Packer, and D. Koller. "Self-paced learning for latent variable models". In: *Advances in neural information processing systems* 23 (2010).

[110] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. "Unsupervised learning of visual features by contrasting cluster assignments". In: *Advances in neural information processing systems* 33 (2020), pp. 9912–9924.