KNOWLEDGE-INTENSIVE AND ENTITY-CENTRIC NATURAL LANGUAGE PROCESSING

By

WENZHENG ZHANG

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Karl Stratos

And approved by

New Brunswick, New Jersey

August 2025

ABSTRACT OF THE DISSERTATION

Knowledge-Intensive and Entity-Centric Natural Language Processing

by WENZHENG ZHANG

Dissertation Director: Karl Stratos

Accessing large-scale external knowledge while maintaining a consistent understanding of real-world entities is essential for modern natural language processing (NLP) systems. This thesis investigates two fundamental capabilities that support this objective: **knowledge-intensive language processing**, which enables models to retrieve and integrate external information, and **entity-centric language understanding**, which facilitates identifying, linking, and reasoning about entities in context.

We first explore knowledge-intensive language processing through the lens of retrieval-based methods. We present a theoretical and empirical analysis of hard negatives in the Noise Contrastive Estimation (NCE) training objective, improve multi-task retrieval by promoting task specialization and propose a retrieval-augmented generation framework that allows models to express their information needs implicitly, eliminating the need for human-specified queries.

Next, we focus on entity-centric language understanding. We introduce a novel approach that reframes entity linking as an inverse open-domain question answering problem, addressing the challenge of predicting mentions without knowing their corresponding entities, and naturally extending NCE to support multi-label retrieval. We also propose a simple yet effective sequence-to-sequence model for coreference resolution, which maps input text to linearized coreference annotations and achieves strong performance with no task-specific model design.

These contributions advance the development of NLP systems that can reason more effectively over external knowledge and entities, enabling stronger performance on a wide range of information-seeking and understanding tasks.

ACKNOWLEDGMENTS

The past five years at Rutgers have been an unforgettable and invaluable journey. I am deeply grateful to all the people who have supported and inspired me along the way.

First and foremost, I extend my deepest thanks to my advisor, Karl Stratos. When I joined Rutgers in 2020, I had a background in Physics and knew very little about computer science, let alone natural language processing. It is truly remarkable that over the past five years, I have been able to conduct research in NLP, and that is entirely due to Karl's mentorship. His patience, knowledge, and unwavering guidance—from teaching me basic Linux commands to helping me shape my research identity—have been instrumental in my development. His brilliance and creativity consistently inspire me; he has an exceptional ability to generate ideas and approach research problems with deep insight. Karl always provided me with the freedom to explore ideas independently while offering thoughtful and constructive feedback. His high standards of research quality and taste have shaped my own aspirations as a researcher, reminding me to follow my curiosity and strive for meaningful contributions. Beyond academics, he has also offered valuable advice on life and career, for which I am immensely thankful.

I would also like to thank the members of my dissertation committee—Hao Wang, Abdeslam Boularias, and Wei Xu—for their time, feedback, and support.

I am grateful to my co-authors, collaborators, and mentors, each of whom taught me so much. I was fortunate to have summer internship opportunities that broadened my research horizons. I sincerely thank Chenyan Xiong and Arnold Overwijk for their generous mentorship at Microsoft in 2022, and Mingda Chen, Victoria Lin, and Scott Yih for their kind guidance at Meta FAIR in 2024.

Many thanks to my fellow students and friends at Rutgers. I thank Siwei Mai, Haizhou Shi, and Xi Chen for encouraging me to stay active and for the many great conversations we shared. I thank Kunpeng Song for both technical and non-technical discussions, and for generously sharing his visitor pass so that my wife could access Meta's Farley building during our internships. I also thank Zhuohuan Li and Tongtong Zhong for many shared stroller walks and fun conversations while we took our babies out together.

I thank my parents, Qigui Zhang and Anyun Hu, for their unwavering encouragement and support. They have always listened patiently to my research updates, even when the topics were unfamiliar. I also thank my sister, Ying Zhang, and my brother-in-law, Zhengyang Tan, for accompanying and caring for my parents when I could not be there in person.

Finally, I express my deepest gratitude to my wife, Yinxia Guo, for her love, patience, and endless support. She has been by my side through every high and low, maintaining a warm and loving home. I am especially grateful to her for taking care of our newborn son, Charles Jiaheng Zhang, almost entirely on her own during the earliest and most demanding months after his birth—while I was juggling the pressures of job hunting and completing my dissertation. Her strength, patience, and devotion made this journey possible for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ANN	Approximate Nearest Neighbor
DocIDs	Document Identifiers
IR	Information Retrieval
KB	Knowledge Base
NCE	Noise Contrastive Estimation
RAG	Retrieval-Augmented Generation
Seq2Seq	Sequence-to-Sequence

CHAPTER 1
INTRODUCTION

1.1 Motivation

Recent advances in pretrained language models have led to impressive performance across a wide range of NLP tasks [1, 2, 3, 4, 5]. However, for tasks that are knowledge-intensive or entity-rich, existing models still struggle to provide reliable and accurate solutions. These tasks require not only access to external knowledge beyond what is encoded in model parameters, but also a deep understanding of entities mentioned in the text and how they interact.

To address these challenges, two complementary paradigms have emerged: knowledge-intensive language processing, which focuses on retrieving and integrating external knowledge [6, 7], and entity-centric language understanding, which focuses on identifying, linking, and reasoning about entities and their references in text [8, 9]. These two components often work in tandem to support complex reasoning and generation in knowledge-rich tasks.

Consider the following example of an open-domain question answering task with contextual input:

> Nikola Tesla and Thomas Edison were both pioneering inventors in the field of electricity. Edison was known for his work on the electric light bulb and direct current (DC) systems, while Tesla championed alternating current (AC). In 1884, Tesla emigrated to the United States and briefly worked for Edison. However, the two had fundamental disagreements over electrical systems. Eventually, he left to pursue his own ideas and began working independently.
>
> **Question:** What did he focus on after leaving?

To answer this question, the model must first perform entity linking to map ”Nikola Tesla” and ”Thomas Edison” to corresponding entries in a knowledge base. It must also resolve the pronoun

”he” in the question via coreference resolution, determining that it refers to Tesla, not Edison. Since the passage itself does not contain enough information to answer the question, the model must then retrieve relevant background knowledge about Tesla’s later work using information retrieval, and finally produce a fluent, grounded response via retrieval-augmented generation.

This example illustrates how entity-centric reasoning and knowledge-intensive processing must be integrated to achieve accurate, interpretable language understanding. This thesis investigates methods that advance both paradigms, with the ultimate goal of building NLP systems that can access external knowledge, resolve ambiguity, and reason coherently about real-world entities.

1.2 Overview

This thesis is organized as follows. Chapter 2 provides background on both knowledge-intensive language processing and entity-centric language understanding.

Part I focuses on knowledge-intensive language processing. We begin with a theoretical analysis of hard negatives in the information retrieval training objective, Noise Contrastive Estimation (NCE) (Chapter 3). We then propose a method to improve multi-task retrieval across multiple knowledge-intensive tasks by promoting task specialty (Chapter 4). Finally, we introduce a query-free Retrieval-Augmented Generation (RAG) system that unifies retrieval and generation, enabling models to implicitly express information needs rather than relying on human-specified queries (Chapter 5).

Part II turns to entity-centric language understanding, focusing on entity linking and coreference resolution. We first present an approach that reformulates entity linking as an inverse open-domain question answering problem, avoiding the dilemma of predicting mentions without knowing their corresponding entities (Chapter 6). We then introduce an extremely simple yet high-performing Sequence-to-Sequence (Seq2Seq) method for coreference resolution, mapping an input document to a tagged sequence that encodes its coreference annotations (Chapter 7).

Finally, Chapter 8 offers concluding remarks and outlines potential directions for future work.

1.3 Contributions

The main contributions of this thesis are summarized as follows:

- **Theoretical analysis of hard negatives in NCE.** We develop analytical tools to study the role of hard negatives in the NCE objective and derive a general score function form that unifies various text retrieval model architectures. We show, both theoretically and empirically, that setting the negative distribution equal to the model distribution reduces bias regardless of the score function used.

- **Multi-label NCE for multi-label retrieval.** We extend the standard NCE framework to a multi-label formulation, enabling retrieval with multiple relevant documents per query. We demonstrate superior performance over alternative multi-label objectives through empirical evaluation on entity linking task.

- **Task-specialized multi-task retrieval.** We show that a single multi-task retriever can outperform task-specific retrievers by promoting task specialization. Our approach combines a well-chosen pretrained base model, task-specific prompting, and an adaptive learning method that encourages parameters to specialize for individual tasks.

- **Query-free retrieval-augmented generation.** We propose a query-free RAG system that integrates retrieval and generation into a unified model that allows the model to express information needs implicitly, without human-specified queries. This query-free RAG approach substantially improves both retrieval and generation performance across multiple knowledge-intensive tasks.

- **Entity linking as inverse open-domain QA.** We reformulate entity linking as an inverse open-domain question answering task, fundamentally resolving the challenge in prior approaches of predicting mentions without knowing their corresponding entities. Our method leverages recent advances in dense retrieval and open-domain QA, removes the dependency on Knowledge Base (KB)-specific mention–candidate dictionaries, works for KBs without such resources, and is data-efficient enough to achieve state-of-the-art results within an academic compute budget.

- **Simple Seq2Seq coreference resolution.** We introduce an extremely simple yet flexible sequence-to-sequence approach to coreference resolution, treating coreference annotation as a target sequence generated from the input text. This method achieves state-of-the-art performance without specialized architectures or extensive hyperparameter tuning.

CHAPTER 2
BACKGROUND

2.1 Knowledge-Intensive Language Processing

Knowledge-intensive language processing refers to tasks whose solutions require access to large external knowledge sources (e.g., Wikipedia or other document collections), rather than relying solely on the parametric knowledge encoded in model weights [6, 7].

For example, in open-domain question answering [10], a system must retrieve relevant passages from a large corpus to answer factoid questions. In fact verification [11], models must gather supporting or refuting evidence from a broad document collection. In entity linking [8], models retrieve candidate entities from a large entity collection and link them to entity mentions identified in the input text.

In this thesis, we focus on two tightly connected components of knowledge-intensive language processing: *Information Retrieval (IR)* from large knowledge sources as an intermediate step, and *Retrieval-Augmented Generation (RAG)* as an end-to-end solution that integrates retrieval with generation.

2.1.1 Information Retrieval

2.1.1.1 Problem Formulation.

Let $\mathcal{Y} = y_1, \ldots, y_N$ be a large corpus consisting of N documents, where N can range from millions to billions. Given a query x (such as a question, a mention context, or a task input), the goal of IR is to efficiently retrieve a small subset of relevant documents, denoted as $\mathrm{TopK}(x) \subset \mathcal{Y}$, that are most relevant to the query x.

2.1.1.2 Existing Approaches

Classical (term-based) IR. In classical information retrieval, both queries x and documents y are represented as high-dimensional but *sparse* vectors over the vocabulary V, where each dimension corresponds to a term. A standard representation is the *tf–idf* weighted vector [12]. For a term t in document y, its weight is

$$w_{t,y} = \mathrm{tf}(t, y) \cdot \mathrm{idf}(t), \tag{2.1}$$

where $\mathrm{tf}(t, y)$ is the frequency of t in y and

$$\mathrm{idf}(t) = \log \frac{N+1}{n_t+1}, \tag{2.2}$$

with N the total number of documents and n_t the number of documents containing t. The similarity between a query x and document y is then often computed via cosine similarity of their tf–idf vectors.

A more effective and widely used scoring function is BM25 [13], which incorporates term frequency saturation and document length normalization:

$$\mathrm{BM25}(x, y) = \sum_{t \in x} \log \frac{N - n_t + 0.5}{n_t + 0.5} \cdot \frac{f(t, y)\,(k_1 + 1)}{f(t, y) + k_1\Big(1 - b + b\frac{|y|}{\mathrm{avgdl}}\Big)}, \tag{2.3}$$

where $f(t, y)$ is the frequency of term t in y, $|y|$ is the document length, avgdl is the average document length in the collection, and k_1, b are hyperparameters.

Term-based methods are efficient, interpretable, and can be implemented at web scale with inverted indexes. However, they rely solely on exact lexical overlap, making them vulnerable to the vocabulary mismatch problem (e.g., “car” vs. “automobile”). This motivates the shift toward *model-based dense retrieval*, which learns semantic representations beyond surface forms, as we will discuss in the next paragraph.

Dense Retrieval. Dense retrieval methods learn *dense* vector representations for both queries x and documents y, enabling similarity computation in a continuous vector space [14, 15, 16]. These methods are typically built on a learnable *Dual-Encoder* architecture [17, 18], where a query encoder $\mathrm{enc}_X^\theta : \mathcal{X} \to \mathbb{R}^d$ and a document encoder $\mathrm{enc}_Y^\theta : \mathcal{Y} \to \mathbb{R}^d$ independently map inputs to a shared embedding space. These encoders are often based on pretrained Transformers [1, 2]; for instance, by extracting the `[CLS]` token from the final layer of BERT [1].

The relevance score between a query x and document y is computed via inner product:

$$\mathrm{s}_\theta(x, y) = \left\langle \mathrm{enc}_X^\theta(x), \mathrm{enc}_Y^\theta(y) \right\rangle . \tag{2.4}$$

The retriever returns the top-k most relevant documents as:

$$\mathrm{Top}k(x) \;=\; \underset{y \in \mathcal{Y}}{\arg\,\text{top-k}}\; \mathrm{s}_\theta(x, y), \tag{2.5}$$

which can be efficiently computed using Approximate Nearest Neighbor (ANN) search over all precomputed document embeddings produced offline by the trained document encoder enc_Y^θ, using libraries such as FAISS [19].

Training the Dual-Encoder modules enc_X^θ and enc_Y^θ typically involves a NCE objective (detailed in Section 2.1.1.3), which encourages high similarity for relevant query-document pairs and low similarity for irrelevant ones.

In addition to standard Dual-Encoder, late-interaction models [20, 21] retain token-level granularity through interaction and aggregation layers, enabling finer-grained semantic alignment at the cost of efficiency. We provide a detailed comparison of score functions in Section 3.4.

Overall, dense retrieval with Dual-Encoder is widely adopted due to its strong balance of efficiency, performance, scalability, flexibility, and generalizability.

Generative Retrieval. For completeness, we briefly review generative IR. Unlike dense retrieval, generative retrieval reformulates the task as a sequence-to-sequence (Seq2Seq) problem: given a query x as input, the model generates Document Identifiers (DocIDs) corresponding to relevant documents [22, 23]. This approach typically involves two stages of training—document-to-DocIDs and query-to-DocIDs—to align queries with relevant documents, which introduces additional training complexity.

While generative IR is conceptually novel and performs well on small-scale knowledge bases, it faces significant challenges. Designing effective *discrete* DocIDs is nontrivial[23, 24, 25, 26], limiting scalability [27]. Furthermore, adapting to dynamic or frequently updated knowledge bases is difficult due to the static nature of tokenized identifiers.

As a result, generative IR has seen limited adoption in large-scale settings, whereas dense retrieval remains the dominant and more practical paradigm. In this thesis, we focus exclusively on dense retrieval, and include generative IR here for completeness.

2.1.1.3 Noise Contrastive Estimation (NCE)

A widely used training objective for dense retrieval models, such as dual-encoder architectures, is Noise Contrastive Estimation (NCE) [28, 29, 30]. NCE trains the model to assign higher scores to positive query-document pairs (x, y_1^+) than to a set of negative documents $y_{2:K}^-$. Without loss of generality, we assume the positive example is always placed in the first position, i.e., $y_1 = y_1^+$.

Given a set of $K \geq 2$ candidates consisting of one positive and $K-1$ negatives, the NCE objective is defined as:

$$J_{\text{NCE}}(\theta) = \mathop{\mathbf{E}}_{\substack{(x,y_1^+)\sim\mathbf{pop} \\ y_{2:K}^-\sim q^{K-1}}} \left[-\log \pi_\theta(y_1^+ | x, y_{1:K}) \right] \tag{2.6}$$

where

$$\pi_\theta(k|x, y_{1:K}) = \frac{\exp\left(s_\theta(x, y_k)\right)}{\sum_{k'=1}^{K} \exp\left(s_\theta(x, y_{k'})\right)} \tag{2.7}$$

and $\mathrm{s}_\theta(x, y)$ denotes the relevance score between query x and document y.

The negatives $y_{2:K}$ are sampled independently from a noise distribution q over the corpus $\mathcal{Y}$. Common choices for q include the uniform distribution $q(y) = 1/|\mathcal{Y}|$ and the marginal distribution $q(y) = \mathcal{P}(y)$.

In practice, it is common to use in-batch negatives[31], which treat the positives of other queries within the same batch as negatives. This improves training efficiency by amortizing negative sampling. Additionally, hard-negative mining[15, 32] is often used to sample more challenging negatives, which can lead to better model performance by increasing training difficulty.

2.1.1.4 Evaluation

The goal of retrieval is to return relevant items for a given query. Depending on the task and application-specific priorities, different evaluation metrics are used to capture various aspects of retrieval quality. Below, we review the most common retrieval metrics: Recall@k, MRR@k, and (N)DCG@k, assuming there are n queries and that k items are retrieved for each query.

Recall@k. Recall@k measures the proportion of relevant items that are successfully retrieved among the top-k candidates. In multi-label or multi-relevant retrieval settings, where each query may have multiple ground-truth relevant items, Recall@k is computed as:

$$\text{Recall@}k = \frac{\sum_{i=1}^{n} |\text{Top}k(x_i) \cap \mathcal{Y}i^{+}|}{\sum_{i=1}^{n} |\mathcal{Y}_i^{+}|}, \tag{2.8}$$

where x_i is the i-th query, $\text{Top}k(x_i)$ is the set of top-k retrieved items, and $\mathcal{Y}_i^{+}$ is the set of ground-truth relevant items for x_i. Recall@k is suitable when coverage is important—i.e., when we care about retrieving as many relevant items as possible, even if they are not ranked at the top.

Mean Reciprocal Rank (MRR@k). MRR evaluates how early the first relevant item appears in the ranking. It is defined as the mean of the reciprocal rank of the first relevant item for each query:

$$\text{MRR} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[\text{rank}_i \leq k] \cdot \frac{1}{\text{rank}_i}, \tag{2.9}$$

where rank_i is the rank position of the first relevant item for query x_i. MRR is especially useful when early precision is important—e.g., when the user is expected to interact only with the top result, such as in FAQ matching or dialogue response retrieval.

Discounted Cumulative Gain (DCG@k) and Normalized DCG (NDCG@k). DCG considers both the relevance and the position of retrieved items, assigning higher weight to relevant items appearing higher in the ranked list. It is defined as:

$$\mathrm{DCG@}k = \sum_{j=1}^{k} \frac{\mathrm{rel}_j}{\log_2(j+1)}, \tag{2.10}$$

where rel_j is the graded relevance (e.g., 0/1 for binary, or a score) of the item at position j. NDCG@k normalizes DCG by the maximum possible DCG (IDCG) for each query:

$$\mathrm{NDCG@}k = \frac{\mathrm{DCG@}k}{\mathrm{IDCG@}k}. \tag{2.11}$$

(N)DCG is appropriate when ranking quality matters—i.e., we care not only about retrieving relevant items but also about ranking them appropriately. This is particularly relevant in web search, product recommendation, or reranking tasks.

2.1.2 Retrieval-Augmented Generation (RAG)

2.1.2.1 Problem Formulation

Retrieval-Augmented Generation (RAG) provides an end-to-end framework for solving knowledge-intensive NLP tasks [7, 33, 34]. It consists of two components: a retriever and a generator. The retriever $r_\theta(y|x)$ selects a small set of relevant documents $y = (y_1, \dots, y_k)$ from a large knowledge base, as described in Section 2.1.1. The generator $g_\phi(a|x, y)$ then produces the final output a conditioned on the original input x and the retrieved documents y. The generator can be instantiated with various pretrained language models, ranging from small-scale sequence-to-sequence models [2, 35] to large-scale LLMs [4, 5].

2.1.2.2 Existing Approaches

Early RAG systems explored two parallel lines of development: pipeline training and joint training. Pipeline approaches [31, 36] train the retriever and generator independently—first training a retriever to fetch relevant documents, followed by training a generator on the fixed retrieved results. In contrast, joint training methods [33, 7] treat the retrieved documents as latent variables and optimize the retriever and generator simultaneously in an end-to-end fashion.

Subsequent research has further advanced the interaction between retrievers and generators by distilling supervision signals from the generator into the retriever. The key observation is that the log-likelihood of producing ground-truth answers increases when conditioned on relevant documents and decreases when conditioned on irrelevant ones. These likelihood-based signals provide a natural training objective for the retriever and have been exploited in multiple ways: through pipeline training [37], iterative instruction tuning [38], joint training [39], or by freezing the generator as a powerful large language model (LLM) and updating only the retriever with the distilled supervision [40].

Another line of work explores adaptive retrieval, where the model dynamically triggers retrieval—potentially multiple times during generation—using different strategies such as prompting [41, 42, 43], supervised learning [44], or reinforcement learning [45].

2.1.2.3 Training

Pipeline Training. In pipeline training, the retriever and generator are trained separately. The retriever is first trained using the Noise Contrastive Estimation (NCE) objective as described in Section 2.1.1.3. Once trained, it is used to retrieve documents for each input query. These fixed documents are then used to train the generator, which learns to generate answers conditioned on both the input and the retrieved documents. The generator is trained with the standard conditional language modeling loss:

$$J_{\text{gen}}(\phi) = - \mathop{\mathbf{E}}_{\substack{(x,a)\sim\mathbf{pop} \\ y_{1:k}\sim r_\theta(y|x)}} \left[\log g_\phi(a \mid x, y_{1:k})\right], \tag{2.12}$$

where (x, a) is a training example with input x and target answer a, and $y_{1:k}$ are the top-k documents retrieved by the fixed retriever.

Joint Training. Joint training optimizes the retriever and generator together by treating the retrieved documents as latent variables. The goal is to maximize the marginal likelihood of the correct answer across all possible retrieved document sets:

$$J_{\text{joint}}(\theta, \phi) = - \mathop{\mathbf{E}}_{(x,a)\sim\mathbf{pop}} \left[\log \sum_{y_{1:k}} r_\theta(y_{1:k} \mid x) \cdot g_\phi(a \mid x, y_{1:k})\right] \tag{2.13}$$

In practice, the summation over all possible document sets is intractable and is approximated using top-k documents retrieved by the current retriever. This approach allows the generator to backpropagate supervision signals to the retriever through the marginal likelihood.

Generator Supervises Retriever. To further enhance retriever quality, the generator's output can be used to distill supervision back to the retriever. While various signals such as attention weights or token probabilities can be used, we focus on perplexity-based distillation, which has shown strong empirical performance and broad applicability.

In this setting, the generator defines a teacher distribution over the retrieved documents based on the likelihood of generating the correct answer:

$$p_\phi^{\text{gen}}(y_i \mid x, a) = \frac{g_\phi(a \mid x, y_i)}{\sum_{j=1}^{k} g_\phi(a \mid x, y_j)}, \tag{2.14}$$

where $g_\phi(a \mid x, y_i)$ is the likelihood of the correct answer given input x and document y_i. The retriever is then trained to match this distribution with its own output distribution $r_\theta(y_i \mid x)$ by minimizing the KL divergence:

$$J_{\text{distill}}(\theta) = \text{KL}\left(\overline{p_\phi^{\text{gen}}(\cdot \mid x, a)} \parallel r_\theta(\cdot \mid x)\right). \tag{2.15}$$

Here, $\overline{p_\phi^{\text{gen}}(\cdot \mid x, a)}$ indicates that gradients are not backpropagated through the target distribution. This encourages the retriever to assign higher probabilities to documents that the generator finds more helpful for answer generation, aligning both components for better end-to-end performance.

2.1.2.4 Evaluation

Since a Retrieval-Augmented Generation (RAG) system consists of both a retriever and a generator, evaluation typically considers both components.

For the retriever, we use the metrics discussed in Section 2.1.1.4, such as Recall@k, MRR, and (N)DCG, which assess how well the retriever selects relevant documents.

For the generator, evaluation depends on the specific knowledge-intensive task. Below, we outline common metrics used for widely studied tasks:

Open-Domain Question Answering. We use Exact Match (EM) to evaluate whether the generated answer exactly matches the ground-truth answer (ignoring case, punctuation, and articles):

$$\text{EM} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[\text{normalize}(a_i^{\text{pred}}) = \text{normalize}(a_i^{\text{gold}})], \tag{2.16}$$

where a_i^{pred} is the generated answer, a_i^{gold} is the ground-truth answer, and normalize$(\cdot)$ removes articles, punctuation, and lowercases the string.

Fact Verification. We use Accuracy to evaluate whether the predicted label (e.g., SUPPORTS, REFUTES, NOT ENOUGH INFO) matches the ground-truth label:

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[\hat{y}_i = y_i], \tag{2.17}$$

where $\hat{y}_i$ is the predicted label and y_i is the true label.

Entity Linking. We evaluate entity linking using Precision, Recall, and F1, computed over predicted and gold mention-entity pairs:

$$\text{Precision} = \frac{|\text{Pred} \cap \text{Gold}|}{|\text{Pred}|},\ \text{Recall} \quad = \frac{|\text{Pred} \cap \text{Gold}|}{|\text{Gold}|},\ \text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.18}$$

Each pair is a tuple `(mention_start, mention_end, entity_name)`, representing the span and the linked entity. A prediction is correct if it exactly matches the gold span and entity.

2.2 Entity-Centric Language Understanding

Entity-centric language understanding refers to the capability of a system to recognize, track, and reason about entities mentioned in text. This includes identifying entity mentions, resolving ambiguities, linking them to structured knowledge bases, and understanding how different mentions refer to the same real-world entity across and within documents. Such understanding is crucial for coherent and grounded natural language understanding, especially in knowledge-rich applications like question answering, summarization, and information extraction.

In this thesis, we focus on two tightly connected core tasks in this domain: *Entity Linking (EL)* and *Coreference Resolution (CR)*. Entity Linking identifies and maps entity mentions in text to corresponding entries in a knowledge base. Coreference Resolution clusters mentions that refer to the same entity, enabling discourse-level understanding.

2.2.1 Entity Linking

2.2.1.1 Problem Formulation

Entity Linking (EL) involves two sub-tasks: (1) identifying potentially ambiguous mention spans in a document that refer to entities, and (2) linking each mention span to the correct entity in a knowledge base (KB). Formally, let $\mathcal{E}$ denote the set of entities in a KB (e.g., Wikipedia articles), and let $\mathcal{X} = x_1, \ldots, x_n$ be the set of documents, each of length up to $T_{\max}$.

The task is to map $x \in \mathcal{X}$ to $y \in \mathcal{P}(\mathcal{Y}(x))$ where $\mathcal{Y}(x) = \{(s, t, e) : 1 \leq s \leq t \leq |x|, e \in \mathcal{E}\}$ is the set of all possible linked spans in x and $\mathcal{P}$ is the power set.

The output space is exponentially large: $O(2^{T_{\max}^2 \cdot |\mathcal{E}|})$. Given that $|\mathcal{E}|$ is typically in the millions (e.g., 6M for Wikipedia) and documents can be long (e.g., $T_{\max} > 3000$ in AIDA), exhaustive search is computationally infeasible.

EL models are evaluated using precision, recall, and F1 over correctly predicted (mention, entity) pairs, as discussed in Section 2.1.2.4.

2.2.1.2 Existing Approaches

Due to the exponential size of the search space, prior work typically decomposes the entity linking task into two subproblems: *Mention Detection (MD)*, which identifies candidate mention spans in the input document, and *Entity Disambiguation (ED)*, which links each detected mention to its corresponding entity in a knowledge base.

Some approaches assume gold or pre-annotated mentions and focus solely on ED [46]. Others adopt a pipeline strategy by first applying an off-the-shelf named entity recognition (NER) system to detect mentions, followed by ED to resolve them (MD $\rightarrow$ED pipeline)[47, 48, 49].

More recent work proposes end-to-end models that jointly perform MD and ED. These models typically use beam search to explore candidate spans and entities, either through embedding-based similarity scoring [8] or by casting the task as a constrained generative retrieval problem [22].

2.2.2 Coreference Resolution

2.2.2.1 *Problem Formulation*

Coreference Resolution (CR) is the task of identifying and clustering textual mentions that refer to the same real-world entity. Formally, let $\mathcal{X} = x_1, \ldots, x_n$ denote a set of documents, each with length up to $T_{\max}$. The goal is to map a document $x \in \mathcal{X}$ to a set $y \in \mathcal{P}(\mathcal{Y}(x))$, where $\mathcal{Y}(x) = \{(s, t, c) : 1 \leq s \leq t \leq |x|, 1 \leq c \leq C\}$ represents all possible clustered mention spans in x. Here, s and t denote the start and end indices of a span, respectively, and c indicates the cluster ID. $\mathcal{P}$ is the power set.

The number of clusters C varies dynamically with each input x, and the number of mentions within each cluster is also variable. Additionally, mention spans can be nested, further increasing the complexity of the task. These structural and combinatorial properties make coreference resolution particularly challenging.

2.2.2.2 *Existing Approaches*

The seminal work by Lee *et al.* [9], followed by a series of improvements [50, 51, 52], established a strong foundation for coreference resolution by computing span-level representations for all possible mention spans in a document. Mention scores are computed via projection layers, while coreference scores for mention pairs are calculated using the dot product of their span embeddings. The model then jointly maximizes the likelihood of gold coreference clusters by combining mention and pairwise coreference scores. To make this approach tractable, various pruning strategies are employed—such as limiting spans to a maximum length L, retaining only the top λT highest-scoring spans, and restricting each mention to at most K candidate antecedents. Although these models are highly effective, they tend to be task-specific and require careful hyperparameter tuning and engineering.

More recent work explores alternative formulations leveraging pretrained sequence-to-sequence models [2, 35]. For example, Liu *et al.* [53] propose an autoregressive, multi-level pointer network for bracket-based generation. Their model predicts mention spans and coreference links using dot product similarity between bracket embeddings, with separate heads for span pairing and coreference resolution. Bohnet *et al.* [54] introduce a transition-based approach, simulating a carefully designed state-action space through a seq2seq model that processes one state at a time. Other efforts attempt to cast coreference resolution directly as a sequence-to-sequence task. Urbizu *et al.* [55] present a proof-of-concept that generates bracketed cluster annotations (e.g., "(0 – – 0) – (1 – (2) — 1)") using a decoder that only outputs symbolic annotations without access to document content, resulting in poor performance (e.g., 66.5 vs. 78.8 B^3 on ARRAU). Similarly, Paolini *et al.* [56] propose translating annotated clusters into target sequences by referencing previous predicted mentions, but fail to achieve competitive results on standard benchmarks.

2.2.2.3 Evaluation

Coreference resolution is typically evaluated using three standard metrics: MUC, B^3, and CEAF$_\phi$, which capture different aspects of clustering performance. The CoNLL-2012 shared task recommends using the average F1 score of these metrics as the final evaluation.

Let M be the set of all mentions in a document. Let $G = \{G_1, G_2, \ldots, G_{|G|}\}$ be the set of gold coreference clusters and $S = \{S_1, S_2, \ldots, S_{|S|}\}$ be the set of predicted (system) clusters. For a mention $m \in M$, let $C_G(m)$ and $C_S(m)$ denote its gold and predicted clusters, respectively.

MUC [57] MUC treats coreference resolution as a link prediction problem.

$$\text{Precision}_{\text{MUC}} = \frac{\sum\limits_{S_i \in S} (|S_i| - |P(S_i)|)}{\sum\limits_{S_i \in S} (|S_i| - 1)} \tag{2.19}$$

$$\text{Recall}_{\text{MUC}} = \frac{\sum\limits_{G_i \in G} (|G_i| - |P(G_i)|)}{\sum\limits_{G_i \in G} (|G_i| - 1)} \tag{2.20}$$

where $P(S_i)$ and $P(G_i)$ are the number of partitions induced by aligning clusters in S and G.

B^3 [58] B^3 computes precision and recall at the mention level:

$$\text{Precision}_{\text{B}^3} = \frac{1}{|M|} \sum_{m \in M} \frac{|C_G(m) \cap C_S(m)|}{|C_S(m)|} \tag{2.21}$$

$$\text{Recall}_{\text{B}^3} = \frac{1}{|M|} \sum_{m \in M} \frac{|C_G(m) \cap C_S(m)|}{|C_G(m)|} \tag{2.22}$$

CEAF$_\phi$ [59] CEAF$_\phi$ finds an optimal 1-1 alignment between predicted and gold clusters based on similarity:

$$\phi(G_i, S_j) = |G_i \cap S_j| \tag{2.23}$$

$$\pi = \arg \max_{\text{1-1 mapping}} \sum_{(G_i, S_{\pi(i)})} \phi(G_i, S_{\pi(i)}) \tag{2.24}$$

$$\text{Precision}_{\text{CEAF}_\phi} = \frac{\sum_{(G_i, S_{\pi(i)})} |G_i \cap S_{\pi(i)}|}{\sum_{S_j \in S} |S_j|} \tag{2.25}$$

$$\text{Recall}_{\text{CEAF}_\phi} = \frac{\sum_{(G_i, S_{\pi(i)})} |G_i \cap S_{\pi(i)}|}{\sum_{G_i \in G} |G_i|} \tag{2.26}$$

F1 Score and CoNLL Score Each metric computes F1 as the harmonic mean:

$$F1 = \frac{2 \cdot P \cdot R}{P + R} \tag{2.27}$$

The overall CoNLL score is the average of the F1 scores from the three metrics:

$$\text{CoNLL F1} = \frac{1}{3} \left(F1_{\text{MUC}} + F1_{\text{B}^3} + F1_{\text{CEAF}_\phi} \right) \tag{2.28}$$

Part I

Knowledge-Intensive Language Processing

CHAPTER 3
HARD NEGATIVES IN NOISE CONTRASTIVE ESTIMATION

This chapter is adapted from joint work with Karl Stratos entitled " Understanding Hard Negatives in Noise Contrastive Estimation" [60].

The choice of negative examples is important in noise contrastive estimation. Recent works find that hard negatives—highest-scoring incorrect examples under the model—are effective in practice, but they are used without a formal justification. We develop analytical tools to understand the role of hard negatives. Specifically, we view the contrastive loss as a biased estimator of the gradient of the cross-entropy loss, and show both theoretically and empirically that setting the negative distribution to be the model distribution results in bias reduction. We also derive a general form of the score function that unifies various architectures used in text retrieval. By combining hard negatives with appropriate score functions, we obtain strong results on the challenging task of zero-shot entity linking.

3.1 Introduction

Noise contrastive estimation (NCE) is a widely used approach to large-scale classification and retrieval. It estimates a score function of input-label pairs by a sampled softmax objective: given a correct pair (x, y_1), choose negative examples $y_2 \dots y_K$ and maximize the probability of (x, y_1) in a softmax over the scores of $(x, y_1) \dots (x, y_K)$. NCE has been successful in many applications, including information retrieval [61], entity linking [15], and open-domain question answering [31].

It is well known that making negatives "hard" can be empirically beneficial. For example, Gillick *et al.* [15] propose a hard negative mining strategy in which highest-scoring incorrect labels under the current model are chosen as negatives. Some works even manually include difficult examples based on external information such as a ranking function [31] or a knowledge base [62].

While it is intuitive that such hard negatives help improve the final model by making the learning task more challenging, they are often used without a formal justification. Existing theoretical results in contrastive learning are not suitable for understanding hard negatives since they focus on unconditional negative distributions [28, 29, 30, 63] or consider a modified loss divergent from practice [64].

In this chapter, we develop analytical tools to understand the role of hard negatives. We formalize hard-negative NCE with a realistic loss (3.6) using a general conditional negative distribution, and view it as a biased estimator of the gradient of the cross-entropy loss. We give a simple analysis of the bias (Theorem 3.3.1). We then consider setting the negative distribution to be the model distribution, which recovers the hard negative mining strategy of Gillick *et al.* [15], and show that it yields an unbiased gradient estimator when the model is optimal (Theorem 3.3.2). We complement the gradient-based perspective with an adversarial formulation (Theorem 3.3.3).

The choice of architecture to parametrize the score function is another key element in NCE. There is a surge of interest in developing efficient cross-attentional architectures [21, 20, 65], but they often address different tasks and lack direct comparisons. We give a single algebraic form of the score function (3.10) that subsumes and generalizes these works, and directly compare a spectrum of architectures it induces.

We present experiments on the challenging task of zero-shot entity linking [66]. We calculate empirical estimates of the bias of the gradient estimator to verify our analysis, and systematically explore the joint space of negative examples and architectures. We have clear practical recommendations: (i) hard negative mining always improves performance for all architectures, and (ii) the sum-of-max encoder [20] yields the best recall in entity retrieval. Our final model combines the sum-of-max retriever with a BERT-based joint reranker to achieve 67.1% unnormalized accuracy: a 4.1% absolute improvement over Wu *et al.* [67]. We also present complementary experiments on AIDA CoNLL-YAGO [47] in which we finetune a Wikipedia-pretrained dual encoder with hard-negative NCE and show a 6% absolute improvement in accuracy.

3.2 Background

3.2.1 Review of NCE

Let $\mathcal{X}$ and $\mathcal{Y}$ denote input and label spaces. We assume $|\mathcal{Y}| < \infty$ for simplicity. Let $\mathbf{pop}$ denote a joint population distribution over $\mathcal{X} \times \mathcal{Y}$. We define a score function $s_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ differentiable in $\theta \in \mathbb{R}^d$. Given sampling access to $\mathbf{pop}$, we wish to estimate θ such that the classifier $x \mapsto \arg\max_{y \in \mathcal{Y}} s_\theta(x, y)$ (breaking ties arbitrarily) has the optimal expected zero-one loss. We can reduce the problem to conditional density estimation. Given $x \in \mathcal{X}$, define

$$p_\theta(y|x) = \frac{\exp\left(s_\theta(x, y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(s_\theta(x, y')\right)} \tag{3.1}$$

for all $y \in \mathcal{Y}$. Let θ^* denote a minimizer of the cross-entropy loss:

$$J_{\text{CE}}(\theta) = \mathop{\mathbf{E}}_{(x,y) \sim \mathbf{pop}} \left[-\log p_\theta(y|x)\right] \tag{3.2}$$

If the score function is sufficiently expressive, θ^* satisfies $p_{\theta^*}(y|x) = \mathbf{pop}(y|x)$ by the usual property of cross entropy. This implies that s_{θ^*} can be used as an optimal classifier.

The cross-entropy loss is difficult to optimize when $\mathcal{Y}$ is large since the normalization term in (3.1) is expensive to calculate. In NCE, we dodge this difficulty by subsampling. Given $x \in \mathcal{X}$ and any K labels $y_{1:K} = (y_1 \ldots y_K) \in \mathcal{Y}^K$, define

$$\pi_\theta(k|x, y_{1:K}) = \frac{\exp\left(s_\theta(x, y_k)\right)}{\sum_{k'=1}^{K} \exp\left(s_\theta(x, y_{k'})\right)} \tag{3.3}$$

for all $1 \leq k \leq K$. When $K \ll |\mathcal{Y}|$, (3.3) is significantly cheaper to calculate than (3.1). Given $K \geq 2$, we define

$$J_{\text{NCE}}(\theta) = \mathop{\mathbf{E}}_{\substack{(x,y_1) \sim \mathbf{pop} \\ y_{2:K} \sim q^{K-1}}} \left[-\log \pi_\theta(1|x, y_{1:K})\right] \tag{3.4}$$

where $y_{2:K} \in \mathcal{Y}^{K-1}$ are negative examples drawn iid from some "noise" distribution q over $\mathcal{Y}$. Popular choices of q include the uniform distribution $q(y) = 1/|\mathcal{Y}|$ and the population marginal $q(y) = \mathbf{pop}(y)$.

The NCE loss (3.4) has been studied extensively. An optimal classifier can be extracted from a minimizer of J_{NCE} [30]; minimizing J_{NCE} can be seen as maximizing a lower bound on the mutual information between $(x, y) \sim \mathbf{pop}$ if q is the population marginal [68]. We refer to Stratos [69] for an overview. However, most of these results focus on unconditional negative examples and do not address hard negatives, which are clearly conditional. We now focus on conditional negative distributions, which are more suitable for describing hard negatives.

3.2.2 Related Work

We discuss related work to better contextualize our contributions. There is a body of work on developing unbiased estimators of the population distribution by modifying NCE. The modifications include learning the normalization term as a model parameter [28, 29] and using a bias-corrected score function [30]. However, they assume unconditional negative distributions and do not explain the benefit of hard negatives in NCE [15, 67, 31, 62]. In contrast, we directly consider the hard-negative NCE loss used in practice (3.6), and justify it as a biased estimator of the gradient of the cross-entropy loss.

Our work is closely related to prior works on estimating the gradient of the cross-entropy loss, again by modifying NCE. They assume the following loss [64], which we will denote by $J_{\text{PRIOR}}(\theta)$:

$$\mathop{\mathbf{E}}_{\substack{(x,y_1)\sim\mathbf{pop} \\ y_{2:K}\sim\nu(\cdot|x,y_1)^K}} \left[-\log \frac{\exp\left(\bar{s}_\theta(x, y_1, y_1)\right)}{\sum_{k=1}^{K} \exp\left(\bar{s}_\theta(x, y_1, y_k)\right)} \right] \tag{3.5}$$

Here, $\nu(\cdot|x, y_1)$ is a conditional distribution over $\mathcal{Y}\backslash\{y_1\}$, and $\bar{s}_\theta(x, y', y)$ is equal to $s_\theta(x, y)$ if $y = y'$ and $s_\theta(x, y) - \log((K-1)\nu(y|x, y_1))$ otherwise. It can be shown that $\nabla J_{\text{PRIOR}}(\theta) = \nabla J_{\text{CE}}(\theta)$ iff $\nu(y|x, y_1) \propto \exp(s_\theta(x, y))$ for all $y \in \mathcal{Y}\backslash\{y_1\}$ [70]. However, (3.5) requires adjusting the score function and iid negative examples, thus less aligned with practice than (3.6). The bias analysis of

$\nabla J_{\text{PRIOR}}(\theta)$ for general $\nu(\cdot|x, y_1)$ is also significantly more complicated than Theorem 3.3.1 [71].

There is a great deal of recent work on unsupervised contrastive learning of image embeddings in computer vision [68, 72, 73, *inter alia*]. Here, $s_\theta(x, y) = E_\theta(x)^\top F_\theta(y)$ is a similarity score between images, and E_θ or F_θ is used to produce useful image representations for downstream tasks. The model is again learned by (3.4) where (x, y_1) are two random corruptions of the same image and $y_{2:K}$ are different images. Robinson *et al.* [74] propose a hard negative distribution in this setting and analyze the behavior of learned embeddings under that distribution. In contrast, our setting is large-scale supervised classification, such as entity linking, and our analysis is concerned with NCE with general hard negative distributions.

In a recent work, Xiong *et al.* [32] consider contrastive learning for text retrieval with hard negatives obtained globally from the whole data with asynchronous updates, as we do in our experiments. They use the framework of importance sampling to argue that hard negatives yield gradients with larger norm, thus smaller variance and faster convergence. However, their argument does not imply our theorems. They also assume a pairwise loss, excluding non-pairwise losses such as (3.4).

3.3 Hard Negatives in NCE

Given $K \geq 2$, we define

$$J_{\text{HARD}}(\theta) = \mathop{\mathbf{E}}_{\substack{(x,y_1)\sim\mathbf{pop} \\ y_{2:K}\sim h(\cdot|x,y_1)}} \left[-\log \pi_\theta(1|x, y_{1:K})\right] \tag{3.6}$$

where $y_{2:K} \in \mathcal{Y}^{K-1}$ are negative examples drawn from a conditional distribution $h(\cdot|x, y_1)$ given $(x, y_1) \sim \mathbf{pop}$. Note that we do not assume $y_{2:K}$ are iid. While simple, this objective captures the essence of using hard negatives in NCE, since the negative examples can arbitrarily condition on the input and the gold (e.g., to be wrong but difficult to distinguish from the gold) and be correlated (e.g., to avoid duplicates).

We give two interpretations of optimizing J_{HARD}. First, we show that the gradient of J_{HARD} is a biased estimator of the gradient of the cross-entropy loss J_{CE}. Thus optimizing J_{HARD} approximates optimizing J_{CE} when we use a gradient-based method, where the error depends on the choice of $h(\cdot|x, y_1)$. Second, we show that the hard negative mining strategy can be recovered by considering an adversarial setting in which $h(\cdot|x, y_1)$ is learned to maximize the loss.

3.3.1 Gradient Estimation

We assume an arbitrary choice of $h(\cdot|x, y_1)$ and $K \geq 2$. Denote the bias at $\theta \in \mathbb{R}^d$ by

$$b(\theta) = \nabla J_{\text{CE}}(\theta) - \nabla J_{\text{HARD}}(\theta)$$

To analyze the bias, the following quantity will be important. For $x \in \mathcal{X}$ define

$$\gamma_\theta(y|x) = \Pr_{\substack{y_1 \sim \mathbf{pop}(\cdot|x) \\ y_{2:K} \sim h(\cdot|x, y_1) \\ k \sim \pi_\theta(\cdot|x, y_{1:K})}} (y_k = y) \tag{3.7}$$

for all $y \in \mathcal{Y}$. That is, $\gamma_\theta(y|x)$ is the probability that y is included as a candidate (either as the gold or a negative) and then selected by the NCE discriminator (3.3).

Theorem 3.3.1. For all $i = 1 \ldots d$,

$$b_i(\theta) = \mathop{\mathbf{E}}_{x \sim \mathbf{pop}} \left[\sum_{y \in \mathcal{Y}} \epsilon_\theta(y|x) \frac{\partial s_\theta(x, y)}{\partial \theta_i} \right]$$

where $\epsilon_\theta(y|x) = p_\theta(y|x) - \gamma_\theta(y|x)$.

Proof. Fix any $x \in \mathcal{X}$ and let $J^x_{\text{CE}}(\theta)$ and $J^x_{\text{HARD}}(\theta)$ denote $J_{\text{CE}}(\theta)$ and $J_{\text{HARD}}(\theta)$ conditioned on x. The difference $J^x_{\text{CE}}(\theta) - J^x_{\text{HARD}}(\theta)$ is

$$\log Z_\theta(x) - \mathop{\mathbf{E}}_{\substack{y_1 \sim \mathbf{pop}(\cdot|x) \\ y_{2:K} \sim h(\cdot|x, y_1)}} [\log Z_\theta(x, y_{1:K})] \tag{3.8}$$

where we define $Z_\theta(x) = \sum_{y' \in \mathcal{Y}} \exp(s_\theta(x, y'))$ and $Z_\theta(x, y_{1:K}) = \sum_{k=1}^{K} \exp(s_\theta(x, y_k))$. For any $(\tilde{x}, \tilde{y})$, the partial derivative of (3.8) with respect to $s_\theta(\tilde{x}, \tilde{y})$ is given by $[[x = \tilde{x}]]\, p_\theta(\tilde{y}|x) - [[x = \tilde{x}]]\, \gamma_\theta(\tilde{y}|x)$ where $[[A]]$ is the indicator function that takes the value 1 if A is true and 0 otherwise. Taking an expectation of their difference over $x \sim \mathbf{pop}$ gives the partial derivative of $b(\theta) = J_{\text{CE}}(\theta) - J_{\text{HARD}}(\theta)$ with respect to $s_\theta(\tilde{x}, \tilde{y})$: $\mathbf{pop}(\tilde{x})(p_\theta(\tilde{y}|\tilde{x}) - \gamma_\theta(\tilde{y}|\tilde{x}))$. The statement follows from the chain rule:

$$b_i(\theta) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \frac{\partial b(\theta)}{\partial s_\theta(x, y)} \frac{\partial s_\theta(x, y)}{\partial \theta_i}$$

□

Theorem 3.3.1 states that the bias vanishes if $\gamma_\theta(y|x)$ matches $p_\theta(y|x)$. Hard negative mining can be seen as an attempt to minimize the bias by defining $h(\cdot|x, y_1)$ in terms of p_θ. Specifically, we define

$$\begin{aligned} & h(y_{2:K}|x, y_1) \\ & \propto [[|\{y_1 \dots y_K\}| = K]] \prod_{k=2}^{K} p_\theta(y_k|x) \end{aligned} \tag{3.9}$$

Thus $h(\cdot|x, y_1)$ has support only on $y_{2:K} \in \mathcal{Y}^{K-1}$ that are distinct and do not contain the gold. Greedy sampling from $h(\cdot|x, y_1)$ corresponds to taking $K - 1$ incorrect label types with highest scores. This coincides with the hard negative mining strategy of Gillick *et al.* [15].

The absence of duplicates in $y_{1:K}$ ensures $J_{\text{CE}}(\theta) = J_{\text{HARD}}(\theta)$ if $K = |\mathcal{Y}|$. This is consistent with (but does not imply) Theorem 3.3.1 since in this case $\gamma_\theta(y|x) = p_\theta(y|x)$. For general $K < |\mathcal{Y}|$, Theorem 3.3.1 still gives a precise bias term. To gain a better insight into its behavior, it is helpful to consider a heuristic approximation given by[1]

$$\gamma_\theta(y|x) \approx \frac{p_\theta(y|x)\exp\left(s_\theta(x,y)\right)}{N_\theta(x)}$$

where $N_\theta(x) = \sum_{y' \in \mathcal{Y}} p_\theta(y'|x)\exp\left(s_\theta(x,y')\right)$. Plugging this approximation in Theorem 3.3.1 we have a simpler equation

$$b_i(\theta) \approx \mathop{\mathbf{E}}_{(x,y)\sim\mathbf{pop}}\left[\left(1-\delta_\theta(x,y)\right)\frac{\partial s_\theta(x,y)}{\partial\theta_i}\right]$$

where $\delta_\theta(x,y) = \exp\left(s_\theta(x,y)\right)/N_\theta(x)$. The expression suggests that the bias becomes smaller as the model improves since $p_\theta(\cdot|x) \approx \mathbf{pop}(\cdot|x)$ implies $\delta_\theta(x,y) \approx 1$ where $(x,y) \sim \mathbf{pop}$.

We can formalize the heuristic argument to prove a desirable property of (3.6): the gradient is unbiased if θ satisfies $p_\theta(y|x) = \mathbf{pop}(y|x)$, assuming iid hard negatives.

Theorem 3.3.2. Assume $K \geq 2$ and the distribution $h(y_{2:K}|x,y_1) = \prod_{k=2}^{K} p_\theta(y_k|x)$ in (3.6). If $p_\theta(y|x) = \mathbf{pop}(y|x)$, then $\nabla J_{\text{HARD}}(\theta) = \nabla J_{\text{CE}}(\theta)$.

Proof. Since $\mathbf{pop}(y|x) = \exp(s_\theta(x,y))/Z_\theta(x)$, the probability $\gamma_\theta(y|x)$ in (3.7) is

$$\begin{aligned}
&\sum_{y_{1:K}\in\mathcal{Y}^K}\prod_{k=1}^{K}\frac{\exp\left(s_\theta(x,y_k)\right)}{Z_\theta(x)}\frac{\exp\left(s_\theta(x,y)\right)}{Z_\theta(x,y_{1:K})}\\
&=\frac{\exp\left(s_\theta(x,y)\right)}{Z_\theta(x)}\sum_{y_{1:K}\in\mathcal{Y}^K}\frac{\prod_{k=1}^{K}\exp\left(s_\theta(x,y_k)\right)}{Z_\theta(x,y_{1:K})}
\end{aligned}$$

The sum marginalizes a product distribution over $y_{1:K}$, thus equals one. Hence $\gamma_\theta(y|x) = p_\theta(y|x)$. The statement follows from Theorem 3.3.1. □

[1]We can rewrite $\gamma_\theta(y|x)$ as

$$\mathop{\mathbf{E}}_{\substack{y_1\sim\mathbf{pop}(\cdot|x)\\ y_{2:K}\sim h(\cdot|x,y_1)}}\left[\frac{\text{count}_{y_{1:K}}(y)\exp\left(s_\theta(x,y)\right)}{\sum_{y'\in\mathcal{Y}}\text{count}_{y_{1:K}}(y')\exp\left(s_\theta(x,y')\right)}\right]$$

where $\text{count}_{y_{1:K}}(y)$ is the number of times y appears in $y_{1:K}$. The approximation uses $\text{count}_{y_{1:K}}(y) \approx p_\theta(y|x)$ under (3.9).

The proof exploits the fact that negative examples are drawn from the model and does not generally hold for other negative distributions (e.g., uniformly random). We empirically verify that hard negatives indeed yield a drastically smaller bias compared to random negatives (Section 3.5.4).

3.3.2 Adversarial Learning

We complement the bias-based view of hard negatives with an adversarial view. We generalize (3.6) and define

$$J_{\text{ADV}}(\theta, h) = \mathop{\mathbf{E}}_{\substack{(x,y_1)\sim\mathbf{pop} \\ y_{2:K}\sim h(\cdot|x,y_1)}} \left[-\log \pi_\theta(1|x, y_{1:K})\right]$$

where we additionally consider the choice of a hard-negative distribution. The premise of adversarial learning is that it is beneficial for θ to consider the worst-case scenario when minimizing this loss. This motivates a nested optimization problem:

$$\min_{\theta\in\mathbb{R}^d} \max_{h\in\mathcal{H}} J_{\text{ADV}}(\theta, h)$$

where $\mathcal{H}$ denotes the class of conditional distributions over $S \subset \mathcal{Y}$ satisfying $|S \cup \{y_1\}| = K$.

Theorem 3.3.3. Fix $\theta \in \mathbb{R}^d$. For any (x, y_1), pick

$$\tilde{y}_{2:K} \in \mathop{\arg\max}_{\substack{y_{2:K}\in\mathcal{Y}^{K-1}: \\ |\{y_1 \ldots y_K\}|=K}} \sum_{k=2}^{K} s_\theta(x, y_k)$$

breaking ties arbitrarily, and define the point-mass distribution over $\mathcal{Y}^{K-1}$:

$$\tilde{h}(y_{2:K}|x, y_1) = [[y_k = \tilde{y}_k \ \forall k = 2 \ldots K]]$$

Then $\tilde{h} \in \arg\max_{h\in\mathcal{H}} J_{\text{ADV}}(\theta, h)$.

Proof. $\max_{h \in \mathcal{H}} J_{\text{ADV}}(\theta, h)$ is equivalent to

$$\max_{h \in \mathcal{H}} \underset{\substack{(x,y_1) \sim \mathbf{pop} \\ y_{2:K} \sim h(\cdot|x,y_1)}}{\mathbf{E}} \left[\log \sum_{k=1}^{K} \exp\left(s_\theta(x, y_k)\right) \right]$$

The expression inside the expectation is maximized by $\tilde{y}_{2:K}$ by the monotonicity of $\log$ and $\exp$, subject to the constraint that $|\{y_1 \dots y_K\}| = K$. $\tilde{h} \in \mathcal{H}$ achieves this maximum. $\square$

3.4 Score Function

Along with the choice of negatives, the choice of the score function $s_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a critical component of NCE in practice. There is a clear trade-off between performance and efficiency in modeling the cross interaction between the input-label pair (x, y). This trade-off spurred many recent works to propose various architectures in search of a sweet spot [21, 65], but they are developed in isolation of one another and difficult to compare. In this section, we give a general algebraic form of the score function that subsumes many of the existing works as special cases.

3.4.1 General Form

We focus on the standard setting in NLP in which $x \in \mathcal{V}^T$ and $y \in \mathcal{V}^{T'}$ are sequences of tokens in a vocabulary $\mathcal{V}$. Let $E(x) \in \mathbb{R}^{H \times T}$ and $F(y) \in \mathbb{R}^{H \times T'}$ denote their encodings, typically obtained from the final layers of separate pretrained transformers like BERT [75]. We follow the convention popularized by BERT and assume the first token is a special symbol (i.e., [CLS]), so that $E_1(x)$ and $F_1(y)$ represent single-vector summaries of x and y. We have the following design choices:

- **Direction**: If $x \to y$, define the query $Q = E(x)$ and key $K = F(y)$. If $y \to x$, define the query $Q = F(y)$ and key $K = E(x)$.
- **Reduction**: Given integers m, m', reduce the number of columns in Q and K to obtain $Q_m \in \mathbb{R}^{H \times m}$ and $K_{m'} \in \mathbb{R}^{H \times m'}$. We can simply select leftmost columns, or introduce an additional layer to perform the reduction.

- **Attention**: Choose a column-wise attention $\mathrm{Attn} : A \mapsto \bar{A}$ either Soft or Hard. If Soft, $\bar{A}_t = \mathrm{softmax}(A_t)$ where the subscript denotes the column index. If Hard, $\bar{A}_t$ is a vector of zeros with exactly one 1 at index $\arg\max_i [A_t]_i$.

Given the design choices, we define the score of (x, y) as

$$s_\theta(x, y) = 1_m^\top Q_m^\top K_{m'} \mathrm{Attn}\left(K_{m'}^\top Q_m\right) \tag{3.10}$$

where 1_m is a vector of m 1s that aggregates query scores. Note that the query embeddings Q_m double as the value embeddings. The parameter vector $\theta \in \mathbb{R}^d$ denotes the parameters of the encoders E, F and the optional reduction layer.

3.4.2 Examples

Dual encoder. Choose either direction $x \to y$ or $y \to x$. Select the leftmost $m = m' = 1$ vectors in Q and K as the query and key. The choice of attention has no effect. This recovers the standard dual encoder used in many retrieval problems [76, 77, 66, 67, 31, 78]: $s_\theta(x, y) = E_1(x)^\top F_1(y)$.

Poly-encoder. Choose the direction $y \to x$. Select the leftmost $m = 1$ vector in $F(y)$ as the query. Choose an integer m' and compute $K_{m'} = E(x)\mathrm{Soft}(E(x)^\top O)$ where $O \in \mathbb{R}^{H \times m'}$ is a learnable parameter ("code" embeddings). Choose soft attention. This recovers the poly-encoder [21]: $s_\theta(x, y) = F_1(y)^\top C_{m'}(x, y)$ where $C_{m'}(x, y) = K_{m'}\mathrm{Soft}\left(K_{m'}^\top F_1(y)\right)$. Similar architectures without length reduction have been used in previous works, for instance the neural attention model of Ganea and Hofmann [79].

Sum-of-max. Choose the direction $x \to y$. Select all $m = T$ and $m' = T'$ vectors in $E(x)$ and $F(y)$ as the query and key. Choose $\mathrm{Attn} = \mathrm{Hard}$. This recovers the sum-of-max encoder (aka., ColBERT) [20]: $s_\theta(x, y) = \sum_{t=1}^{T} \max_{t'=1}^{T'} E_t(x)^\top F_{t'}(y)$.

Multi-vector. Choose the direction $x \to y$. Select the leftmost $m = 1$ and $m' = 8$ vectors in $E(x)$ and $F(y)$ as the query and key. Choose $\mathrm{Attn} = \mathrm{Hard}$. This recovers the multi-vector encoder [65]: $s_\theta(x, y) = \max_{t'=1}^{m'} E_1(x)^\top F_{t'}(y)$. It reduces computation to fast dot products over cached embeddings, but is less expressive than the sum-of-max.

The abstraction (3.10) is useful because it generates a spectrum of architectures as well as unifying existing ones. For instance, it is natural to ask if we can further improve the poly-encoder by using $m > 1$ query vectors. We explore these questions in experiments.

3.5 Experiments

We now study empirical aspects of the hard-negative NCE (Section 3.3) and the spectrum of score functions (Section 3.4). Our main testbed is Zeshel [66], a challenging dataset for zero-shot entity linking. We also present complementary experiments on AIDA CoNLL-YAGO [47].[2]

3.5.1 Task

Zeshel contains 16 domains (fictional worlds like *Star Wars*) partitioned to 8 training and 4 validation and test domains. Each domain has tens of thousands of entities along with their textual descriptions, which contain references to other entities in the domain and double as labeled mentions. The input x is a contextual mention and the label y is the description of the referenced entity. A score function $s_\theta(x, y)$ is learned in the training domains and applied to a new domain for classification and retrieval. Thus the model must read descriptions of unseen entities and still make correct predictions.

We follow prior works and report micro-averaged top-64 recall and macro-averaged accuracy for evaluation. The original Zeshel paper [66] distinguishes normalized vs unnormalized accuracy. Normalized accuracy assumes the presence of an external retriever and considers a mention only if its gold entity is included in top-64 candidates from the retriever. In this case, the problem is

[2]Our code is available at: https://github.com/WenzhengZhang/hard-nce-el.

reduced to reranking and a computationally expensive joint encoder can be used. Unnormalized accuracy considers all mentions. Our goal is to improve unnormalized accuracy.

Logeswaran *et al.* [66] use BM25 for retrieval, which upper bounds unnormalized accuracy by its poor recall (first row of Table 3.1). Wu *et al.* [67] propose a two-stage approach in which a dual encoder is trained by hard-negative NCE and held fixed, then a BERT-based joint encoder is trained to rerank the candidates retrieved by the dual encoder. This approach gives considerable improvement in unnormalized accuracy, primarily due to the better recall of a trained dual encoder over BM25 (second row of Table 3.1). We show that we can further push the recall by optimizing the choice of hard negatives and architectures.

3.5.2 Architectures

We represent x and y as length-128 wordpiece sequences where the leftmost token is the special symbol [CLS]; we mark the boundaries of a mention span in x with special symbols. We use two independent BERT-bases to calculate mention embeddings $E(x) \in \mathbb{R}^{768\times 128}$ and entity embeddings $F(y) \in \mathbb{R}^{768\times 128}$, where the columns $E_t(x), F_t(y)$ are contextual embeddings of the t-th tokens.

Retriever. The retriever defines $s_\theta(x, y)$, the score between a mention x and an entity y, by one of the architectures described in Section 3.4.2:

$$E_1(x)^\top F_1(y) \qquad \text{(DUAL)}$$

$$F_1(y)^\top C_m(x, y) \qquad \text{(POLY-}m\text{)}$$

$$\max_{t=1}^{m} E_1(x)^\top F_t(y) \qquad \text{(MULTI-}m\text{)}$$

$$\sum_{t=1}^{128} \max_{t'=1}^{128} E_t(x)^\top F_{t'}(y) \qquad \text{(SOM)}$$

denoting the dual encoder, the poly-encoder [21], the multi-vector encoder [65], and the sum-of-max encoder [20]. These architectures are sufficiently efficient to calculate $s_\theta(x, y)$ for all entities y in training domains for each mention x. This efficiency is necessary for sampling hard negatives during training and retrieving candidates at test time.

Reranker. The reranker defines $s_\theta(x,y) = w^\top E_1(x,y) + b$ where $E(x,y) \in \mathbb{R}^{H\times 256}$ is BERT (either base $H = 768$ or large $H = 1024$) embeddings of the concatenation of x and y separated by the special symbol [SEP], and w, b are parameters of a linear layer. We denote this encoder by JOINT.

3.5.3 Optimization

Training a retriever. A retriever is trained by minimizing an empirical estimate of the hard-negative NCE loss (3.6),

$$\widehat{J}_{\text{HARD}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left(s_\theta(x_i, y_{i,1})\right)}{\sum_{k'=1}^{K} \exp\left(s_\theta(x_i, y_{i,k'})\right)} \tag{3.11}$$

where $(x_1, y_{1,1}) \dots (x_N, y_{N,1})$ denote N mention-entity pairs in training data, and $y_{i,2} \dots y_{i,K} \sim h(\cdot|x_i, y_{i,1})$ are $K-1$ negative entities for the i-th mention. We vary the choice of negatives as follows.

- Random: The negatives are sampled uniformly at random from all entities in training data.
- Hard: The negatives are sampled from (3.9) each epoch. That is, in the beginning of each training pass, for each i we sample entities $y_{i,2} \dots y_{i,K}$ from $\mathcal{Y} \backslash \{y_{i,1}\}$ without replacement with probabilities proportional to $\exp\left(s_\theta(x_i, y_{i,k})\right)$. This is slightly different from, and simpler than, the original hard negative mining strategy of Gillick *et al.* [15] which pretrains the model using random negatives then greedily adds negative entities that score higher than the gold.
- Mixed-p: p percent of the negatives are hard, the rest are random. Previous works have shown that such a combination of random and hard negatives can be effective. We find the performance is not sensitive to the value of p.

We experimented with in-batch sampling as done in previous works (e.g., Gillick *et al.* [15]), but found sampling from all training data to be as effective and more straightforward (e.g., the number

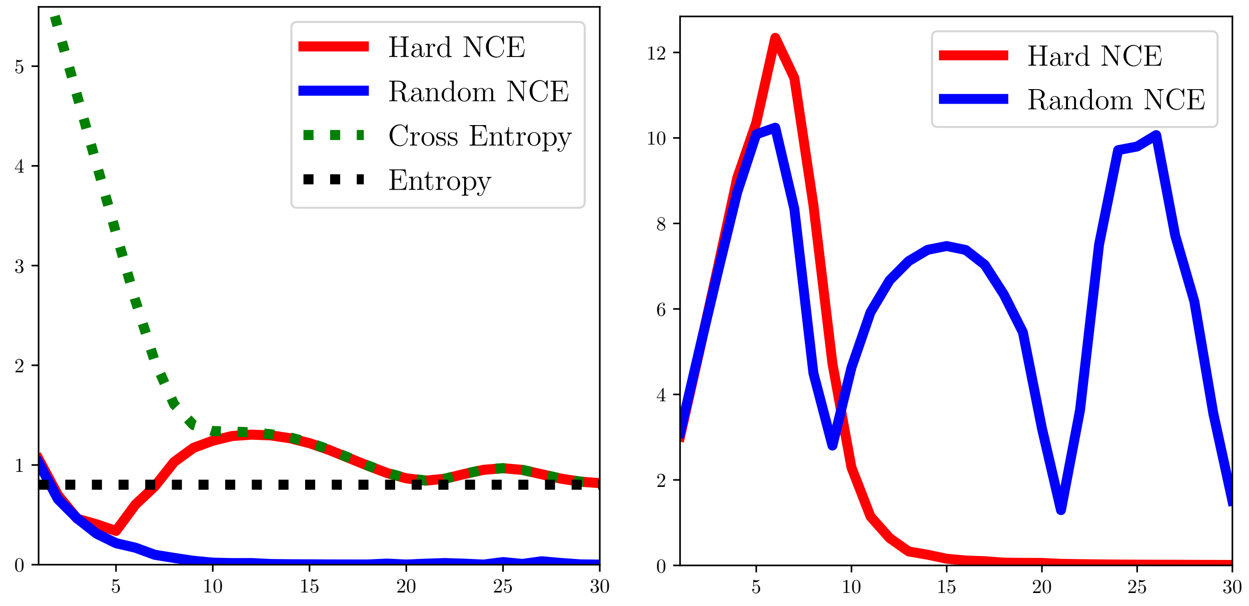


Figure 3.1: Synthetic experiments. We use a feedforward network to estimate the population distribution by minimizing sampled cross entropy in each step (x-axis). We show the NCE loss (left) and the norm of the gradient bias (right) using hard vs random negatives.

of random negatives is explicitly unrelated to the batch size). We use $K = 64$ in all experiments.

Training a reranker. We use JOINT only for reranking by minimizing (3.11) with top-63 negatives given by a fixed retriever, where we vary the choice of retriever. We also investigate other architectures for reranking such as the poly-encoder and the sum-of-max encoder, but we find the full cross attention of JOINT to be indispensable.

Other details. All models are trained up to 4 epochs using Adam. We tune the learning rate over $\{5\text{e}{-5}, 2\text{e}{-5}, 1\text{e}{-5}\}$ on validation data. We use the training batch size of 4 mentions for all models except for JOINT, for which we use 2. Training time is roughly half a day on a single NVIDIA A100 GPU for all models, except the SOM retriever which takes 1-2 days.

Model	Negatives	Val	Test
BM25	–	76.22	69.13
Wu *et al.* [67]	Mixed (10 hard)	91.44	82.06
DUAL	Random	91.08	81.80
	Hard	91.99	84.87
	Mixed-50	91.75	84.16
DUAL-(3.5)	Hard	91.57	83.08
POLY-16	Random	91.05	81.73
	Hard	92.08	84.07
	Mixed-50	92.18	84.34
MULTI-8	Random	91.13	82.44
	Hard	92.35	84.94
	Mixed-50	92.76	84.11
SOM	Random	92.51	87.62
	Hard	94.49	88.68
	Mixed-50	**94.66**	**89.62**

Table 3.1: Top-64 recalls over different choices of architecture and negative examples for a retriever trained by NCE. Wu *et al.* [67] train a dual encoder by NCE with 10 hard negatives. DUAL-(3.5) is DUAL trained with the score-adjusted loss (3.5).

3.5.4 Bias

We conduct experiments on synthetic data to empirically validate our bias analysis in Section 3.3.1.

We construct a population distribution over 1000 labels with small entropy to represent the peaky conditional label distribution $\mathbf{pop}(y|x)$. We use a feedforward network with one ReLU layer to estimate this distribution by minimizing the empirical cross-entropy loss based on 128 iid samples per update. At each update, we compute cross-entropy (3.2) exactly, and estimate NCE (3.6) with 4 negative samples by Monte Carlo (10 simulations).

Figure 3.1 plots the value of the loss function (left) and the norm of the gradient bias (right) across updates. We first observe that hard NCE yields an accurate estimate of cross entropy even with 4 samples. In contrast, random NCE quickly converges to zero, reflecting the fact that the model can trivially discriminate between the gold and random labels. We next observe that the bias of the gradient of hard NCE vanishes as the model distribution converges to the population distribution, which supports our analysis that the bias becomes smaller as the model improves. The bias remains nonzero for random NCE.

Model	**Retriever**	**Negatives**	**Joint Reranker**	**Unnormalized**	
				Val	**Test**
Logeswaran *et al.* [66]	BM25	–	base	–	55.08
Logeswaran *et al.* [66]+DAP	BM25	–	base	–	55.88
Wu *et al.* [67]	DUAL (base)	Mixed (10 hard)	base	–	61.34
Wu *et al.* [67]	DUAL (base)	Mixed (10 hard)	large	–	63.03
Ours	DUAL (base)	Hard	base	69.14	65.42
	DUAL (base)	Hard	large	68.31	65.32
	SOM (base)	Hard	base	69.19	66.67
	SOM (base)	Hard	large	70.08	65.95
	SOM (base)	Mixed-50	base	69.22	65.37
	SOM (base)	Mixed-50	large	**70.28**	**67.14**

Table 3.2: Unnormalized accuracies with two-stage training. DAP refers to domain adaptive pre-training on source and target domains.

Mention	. . . his temporary usurpation of the Imperial throne by invading and seized control of the Battlespire, the purpose of this being to cripple the capacity of the Imperial College of Battlemages, which presented a threat to Tharn's power as Emperor. Mehrunes Dagon was responsible for the **destruction** of Mournhold at the end of the First Era, and apparently also . . .
Random	1. **Mehrunes Dagon** is one of the seventeen Daedric Princes of Oblivion and the primary antagonist of . . .
	2. **Daedric Forces of Destruction** were Mehrunes Dagon's personal army, hailing from his realm of Oblivion, the Deadlands. . . .
	3. **Weir Gate** is a device used to travel to Battlespire from Tamriel. During the Invasion of the Battlespire, Mehrunes Dagon's forces . . .
	4. **Jagar Tharn** was an Imperial Battlemage and personal adviser to Emperor Uriel Septim VII. Tharn used the Staff of Chaos . . .
	5. **House Sotha** was one of the minor Houses of Vvardenfell until its destruction by Mehrunes Dagon in the times of Indoril Nerevar. . . .
	6. **Imperial Battlespire** was an academy for training of the Battlemages of the Imperial Legion. The Battlespire was moored in . . .
Hard	1. **Fall of Ald'ruhn** was a battle during the Oblivion Crisis. It is one of the winning battles invading in the name of Mehrunes Dagon . . .
	2. **Daedric Forces of Destruction** were Mehrunes Dagon's personal army, hailing from his realm of Oblivion, the Deadlands. . . .
	3. **House Sotha** was one of the minor Houses of Vvardenfell until its destruction by Mehrunes Dagon in the times of Indoril Nerevar. . . .
	✓4. **Sack of Mournhold** was an event that occurred during the First Era. It was caused by the Dunmer witch Turala Skeffington . . .
	5. **Mehrunes Dagon of the House of Troubles** is a Tribunal Temple quest, available to the Nerevarine in . . .
	6. **Oblivion Crisis**, also known as the Great Anguish to the Altmer or the Time of Gates by Mankar Camoran, was a period of major turmoil . . .

Table 3.3: A retrieval example with hard negative training on Zeshel. We use a SOM retriever trained with random vs hard negatives (92.51 vs 94.66 in top-64 validation recall). We show a validation mention (**destruction**) whose gold entity is retrieved by the hard-negative model but not by the random-negative model. Top entities are shown for each model (title boldfaced); the correct entity is **Sack of Mournhold** (checkmarked).

3.5.5 Retrieval

Table 3.1 shows the top-64 recall (i.e., the percentage of mentions whose gold entity is included in the 64 entities with highest scores under a retriever trained by (3.6)) as we vary architectures and negative examples. We observe that hard and mixed negative examples always yield sizable improvements over random negatives, for all architectures. Our dual encoder substantially outperforms the previous dual encoder recall by Wu *et al.* [67], likely due to better optimization such as global vs in-batch random negatives and the proportion of hard negatives. We also train a dual encoder with the bias-corrected loss (3.5) and find that this does not improve recall. The poly-encoder and the multi-vector models are comparable to but do not improve over the dual encoder. However, the sum-of-max encoder delivers a decisive improvement, especially with hard negatives, pushing the test recall to above 89%. Based on this finding, we use DUAL and SOM for retrieval in later experiments.

3.5.6 Results

We show our main results in Table 3.2. Following Wu *et al.* [67], we do two-stage training in which we train a DUAL or SOM retriever with hard-negative NCE and train a JOINT reranker to rerank its top-64 candidates. All our models outperform the previous best accuracy of 63.03% by Wu *et al.* [67]. In fact, our dual encoder retriever using a BERT-base reranker outperforms the dual encoder retriever using a BERT-large reranker (65.42% vs 63.03%). We obtain a clear improvement by switching the retriever from dual encoder to sum-of-max due to its high recall (Table 3.1). Using a sum-of-max retriever trained with mixed negatives and a BERT-large reranker gives the best result 67.14%.

3.5.7 Qualitative Analysis

To better understand practical implications of hard negative mining, we compare a SOM retriever trained on Zeshel with random vs hard negatives (92.51 vs 94.66 in top-64 validation recall). The mention categories most frequently improved are Low Overlap (174 mentions) and Multiple Categories (76 mentions) (see Logeswaran *et al.* [66] for the definition of these categories), indicating that hard negative mining makes the model less reliant on string matching. A typical example of improvement is shown in Table 3.3. The random-negative model retrieves person, device, or institution entities because they have more string overlap (e.g. "Mehrunes Dagon", "Battlespire", and "Tharn"). In contrast, the hard-negative model appears to better understand that the mention is referring to a chaotic event like the Fall of Ald'ruhn, Sack of Mournhold, and Oblivion Crisis and rely less on string matching. We hypothesize that this happens because string matching is sufficient to make a correct prediction during training if negative examples are random, but insufficient when

Model	Accuracy
BLINK without finetuning	80.27
BLINK with finetuning	81.54
DUAL with $p = 0$	82.40
DUAL with $p = 50$	88.01
MULTI-2 with $p = 50$	88.39
MULTI-3 with $p = 50$	87.94

Table 3.4: Test accuracies on AIDA CoNLL-YAGO. BLINK refers to the two-stage model of Wu *et al.* [67] pretrained on Wikipedia. All our models are initialized from the BLINK dual encoder and finetuned using all 5.9 million Wikipedia entities as candidates.

they are hard.

To examine the effect of encoder architecture, we also compare a DUAL vs SOM retriever both trained with mixed negatives (91.75 vs 94.66 in top-64 validation recall). The mention categories most frequently improved are again Low Overlap (335 mentions) and Multiple Categories (41 mentions). This indicates that cross attention likewise helps the model less dependent on simple string matching, presumably by allowing for a more expressive class of score functions.

3.5.8 Results on AIDA

We complement our results on Zeshel with additional experiments on AIDA. We use BLINK, a Wikipedia-pretrained two-stage model (a dual encoder retriever pipelined with a joint reranker, both based on BERT) made available by Wu *et al.* [67].[3] We extract the dual encoder module from BLINK and finetune it on AIDA using the training portion. During finetuning, we use all 5.9 million Wikipedia entities as candidates to be consistent with prior work. Because of the large scale of the knowledge base we do not consider SOM and focus on the MULTI-m retriever (DUAL is a special case with $m = 1$). At test time, all models consider all Wikipedia entities as candidates. For both AIDA and the Wikipedia dump, we use the version prepared by the KILT benchmark [6].

Table 3.4 shows the results. Since Wu *et al.* [67] do not report AIDA results, we take the performance of BLINK without and with finetuning from their GitHub repository and the KILT

[3]https://github.com/facebookresearch/BLINK

leaderboard.[4] We obtain substantially higher accuracy by mixed-negative training even without reranking.[5] There is no significant improvement from using $m > 1$ in the multi-vector encoder on this task.

3.6 Conclusions

Hard negatives can often improve NCE in practice, substantially so for entity linking [15], but are used without justification. We have formalized the role of hard negatives in quantifying the bias of the gradient of the contrastive loss with respect to the gradient of the full cross-entropy loss. By jointly optimizing the choice of hard negatives and architectures, we have obtained new state-of-the-art results on the challenging Zeshel dataset [66].

[4]https://ai.facebook.com/tools/kilt/ (as of April 8, 2021)

[5]We find that reranking does not improve accuracy on this task, likely because the task does not require as much reading comprehension as Zeshel.

CHAPTER 4
PROMOTING TASK SPECIALIZATION FOR MULTI-TASK RETRIEVAL

This chapter is adapted from joint work with Chenyan Xiong, Karl Stratos and Arnold Overwijk entitled " Improving Multitask Retrieval by Promoting Task Specialization" [80].

In multitask retrieval, a single retriever is trained to retrieve relevant contexts for multiple tasks. Despite its practical appeal, naive multitask retrieval lags behind task-specific retrieval in which a separate retriever is trained for each task. We show that it is possible to train a multitask retriever that outperforms task-specific retrievers by promoting task specialization. The main ingredients are: (1) a better choice of pretrained model—one that is explicitly optimized for multitasking—along with compatible prompting, and (2) a novel adaptive learning method that encourages each parameter to specialize in a particular task. The resulting multitask retriever is highly performant on the KILT benchmark. Upon analysis, we find that the model indeed learns parameters that are more task-specialized compared to naive multitasking without prompting or adaptive learning.

4.1 Introduction

A standard approach to knowledge-intensive language tasks such as question answering (QA), entity disambigution, and fact verification is retrieval-based. Given an query, a retriever is used to efficiently search a large knowledge base (KB) to retrieve relevant "contexts", typically in the form of short paragraphs. How these contexts are used is task-specific (e.g., entity disambiguation takes the title of the article in which the top retrieved context is found; QA predicts an answer from the contexts by through a reader model). In this paper, we focus on the retrieval step.

In particular, we focus on multitask retrieval. In this setting, there are $K > 1$ downstream tasks that benefit from retrieval from a shared KB. A single retriever is then tasked with performing retrieval for K tasks. Multitask retrieval contrasts with task-specific retrieval in which a separate retriever is trained for each task, and has compelling advantages such as model simplicity (i.e., we can use the same model for all tasks rather than having to design potentially different models for different tasks) and memory efficiency at test time (K times smaller).

Despite the practical appeal, the performance of multitask retrieval has been underwhelming, severely limiting its real-world applicability. Specifically, previous work by Maillard *et al.* [81] trains DPR [14] on the union of all training datasets in the KILT benchmark [6], but the model is outperformed by task-specific retrievers in 5 out of 8 tasks (page-level R-precision, validation split). In our experiments, we find that it is in fact outperformed in all tasks (often by substantial margins) when a stronger task-specific baseline is used. This result is surprising as well as disappointing given the usual benefits of multitask learning (e.g., data efficiency, reduced overfitting) when properly done.

We debunk the previous negative result by presenting a multitask retriever that outperforms task-specific retrievers. The main theme of our work is that it is beneficial to explicitly promote task specialization. A first important source of improvement is a better choice of pretrained model, one that is explicitly optimized for multitasking. Specifically, instead of the standard retrieval encoder BERT [1], we use T5 [2] which includes multitasking in its pretraining stage. Importantly, we use the same prompting as in pretraining (i.e., task indicator) to reduce the gap between pretraining and finetuning for multitask retrieval. A second source of improvement is a novel adaptive learning method in which we adatively upweight the task gradients by the parameter's sensitivity to these tasks to encourage task specialization.

The resulting multitask retriever is highly performant on the KILT benchmark. We achieve 73.74% average page-level R-precision on KILT validation data and 72.84% average page-level R-precision on KILT test data. Upon analysis, we find that the model indeed learns parameters that are more task-specialized compared to naive multitasking without prompting or adaptive learning.

4.2 Related Work

Maillard *et al.* [81] propose multitask retrieval largely as an extension of DPR. Their best model is a BERT-based dual encoder trained on the union of 8 retrieval tasks. While it performs comparably with task-specific DPRs on some tasks, it generally lags behind. In this chapter, we use stronger task-specific retrievers based on T5 and ANCE [82] all of which substantially outperform their multitask retriever. We argue that this negative result undermines the case for multitask retrieval and that it is crucial to demonstrate competitive performance. Our main contribution is producing this demonstration.

We emphasize that achieving competitive multitask retrieval in practice is a highly difficult empirical problem. One might think that it is simply an application of multitask learning, which has no shortage of sophisticated techniques. These techniques typically modify the gradients during training, such as gradient surgery [83], gradient vaccine [84], common gradient descent [85], and GradNorm [86]. We experiment with these techniques and find that they do not help, thus motivated to develop one that works.

Our technical contribution is a new method for multitask learning based on the notion of task sensitivity. Given a loss function $J(\theta)$, the sensitivity of the i-th parameter to the loss at θ is defined as the absolute change in the loss when θ_i is set to zero, which can be approximated by a first-order Taylor approximation as

$$|J(\theta) - J(\theta_{-i})| \approx \left| \frac{\partial J(\theta)}{\partial \theta_i} \times \theta_i \right|$$

where θ_{-i} is equal to θ except that its i-th element is zero. This quantity has been used in the context of model pruning—as a way of identifying weakly sensitive weights [87, 88, 89, 90] and updating them more aggresively [91]. In contrast, we use the quantity to identify weights that are strongly sensitive to a particular task and increase their sensitivity even further, intuitively to achieve per-parameter task specialization. To our knowledge, we are the first to use parameter sensitivity for multitasking learning.

We briefly differentiate our work from other recent works on multitask retrieval. Chen *et al.* [92] present CorpusBrain, an autoregressive multitask retriever trained in largely the same style as GENRE [22] with excellent performance. Autoregressive retrieval has different pros and cons compared to dense retrieval which is our setting; it can be more memory and runtime efficient, but it does not "read" the description of the target and thus not suited for retrieval tasks that require involved reasoning over query-target pairs (e.g., zero-shot entity retrieval [66]). Thus we consider the contribution of CorpusBrain to be at least partially orthogonal to ours. Nevertheless, we show that our model outperforms CorpusBrain in a similar training setting in experiments. Asai *et al.* [93] propose instruction-based retrieval in which the retriever is given an intent as well as a query to find the intended target. While this is a form of multitask retrieval, the problem formulation is different and it is evaluated on its own dataset benchmark.

4.3 Method

We build on the well-established framework of dual encoder [94, 61, 15, 14, *inter alia*]. Let $\mathcal{X}$ denote the set of all queries and $\mathcal{Y}$ the set of all targets (i.e., KB). First, we assume mappings $\mathrm{text}_X : \mathcal{X} \to \mathcal{V}^+$ and $\mathrm{text}_Y : \mathcal{Y} \to \mathcal{V}^+$ where $\mathcal{V}$ denotes the vocabulary to "verbalize" queries and targets. Second, we assume encoders $\mathrm{enc}_X^\theta, \mathrm{enc}_Y^\theta : \mathcal{V}^+ \to \mathbb{R}^d$ with parameters θ defining the relevance score function $\mathrm{s}_\theta(x, y) = \left\langle \mathrm{enc}_X^\theta(\mathrm{text}_X(x)), \mathrm{enc}_Y^\theta(\mathrm{text}_Y(y)) \right\rangle$. Third, assuming iid samples $(x_1, y_1) \dots (x_N, y_N) \sim \mathbf{pop}$, we learn the parameters by noise contrastive estimation (NCE):

$$\min_\theta \; -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(\mathrm{s}_\theta(x_i, y_i))}{\sum_{y \in \mathcal{Y}_i} \exp(\mathrm{s}_\theta(x_i, y))}$$

where $\mathcal{Y}_i \subset \mathcal{Y}$ satisfying $y_i \in \mathcal{Y}_i$ is a set containing the gold and negative targets for the i-th labeled example. We pre-encode every $y \in \mathcal{Y}$ to $v_y = \mathrm{enc}_Y^\theta(\mathrm{text}_Y(y))$ at test time and efficiently compute the highest scoring target $\hat{y}(x) = \arg\max_{y \in \mathcal{Y}} \left\langle \mathrm{enc}_X^\theta(\mathrm{text}_X(x)), v_y \right\rangle$ for any $x \in \mathcal{X}$ by maximum inner product search.

In multitask retrieval, there are K retrieval tasks each with N_k training examples $(x_1^{(k)}, y_1^{(k)}) \dots (x_{N_k}^{(k)}, y_{N_k}^{(k)}) \sim \mathbf{pop}_k$ drawn iid from the k-th population distribution $\mathbf{pop}_k$. We use the KILT benchmark which includes $K = 8$ tasks addressing QA, entity linking, fact checking, slot filling, and dialogue.[1] The per-task loss is

$$J_k(\theta) = -\frac{1}{N_k}\sum_{i=1}^{N_k} \log \frac{\exp(\mathrm{s}_\theta(x_i^{(k)}, y_i^{(k)}))}{\sum_{y \in \mathcal{Y}_i^{(k)}} \exp(\mathrm{s}_\theta(x_i^{(k)}, y))}$$

defining the final loss

$$J(\theta) = \sum_{k=1}^{K} \frac{N_k}{N} \times J_k(\theta)$$

Previous work by Maillard *et al.* [81] use the following setting. The KB $\mathcal{Y}$ consists of 100-token disjoint Wikipedia passages. The text mappings $\mathrm{text}_X, \mathrm{text}_Y$ apply the BERT tokenizer to unmodified queries and passages. The encoders $\mathrm{enc}_X^\theta, \mathrm{enc}_Y^\theta$ are initialized with independent pretrained BERT-bases (uncased). The task-specific training datasets are downsampled to be of similar sizes. As in DPR, they train the model using hard negatives based on BM25, followed by one round of hard negative mining from the model (only on Natural Questions and TriviaQA in which verifying if a candidate negative is indeed incorrect is expedient).

We now describe the main sources of improvement that we achieve over the baseline multitask retriever: a better choice of the base model with appropriate prompting, and better optimization.

4.3.1 Base Model

We use a shared T5 to parameterize and initialize the query and passage encoder $\mathrm{enc}^\theta = \mathrm{enc}_X^\theta = \mathrm{enc}_Y^\theta$. Specifically, we follow the ST5-EncDec architecture [95] which encode any $z \in \mathcal{V}^+$ as

$$\mathrm{enc}^\theta(z) = \mathrm{T5.generate}(z,\ \mathrm{length} = 1).\mathrm{state}$$

(i.e., we run the T5-encoder on z, run the T5-decoder for 1 step from the special start symbol, and take the resulting hidden state prior to token prediction). In addition, we define the text mapping

[1]We write “task” and “dataset” synonymously instead of distinguishing datasets from task types as done in some previous works. Thus KILT has 8 tasks and 5 task types.

for queries $x \in \mathcal{X}$ in task k as

$$\text{text}_X(x) = \text{T5Tokenizer}(\pi_k \oplus \text{[SEP]} \oplus x)$$

where $\oplus$ is string concatenation, [SEP] is the special separation token, and π_k is a text prefix that indicates which task x is a query of. We use dataset names as prefixes (e.g., $\pi_1 =$"NQ"). The text mapping for passages $y \in \mathcal{Y}$ does not use prefixes, that is

$$\text{text}_Y(y) = \text{T5Tokenizer}(y)$$

This allows us to pre-encode passage embeddings at test time and retain the efficiency of the single-task dual encoder framework.

While simple, this choice is the most crucial component in our apporach to improving multitask retrieval. We take a model pretrained for multitasking and adopt the same prefix concatenation scheme for task adaptation, treating multitask retrieval as a continuation of the T5 training.

Interestingly, using task markers is reported to be not helpful in Maillard *et al.* [81]. This is likely because their base model, BERT, is not pretrained for multitasking. Another difference is that they use task markers to indicate the 5 task types (e.g., "QA"), whereas we use fine-grained markers to indicate the 8 tasks (e.g., "NQ"). While there are previous works that use T5 for dense retrieval [95], we are the first to exploit the multitasking component of T5 pretraining for multitask retrieval.

4.3.2 Adaptive Learning

For the k-th task, the linear approximation of $J_k(\theta)$ around $a \in \mathbb{R}^d$ is

$$J_k(\theta) \approx J_k(a) + \langle \nabla J_k(a), \theta - a \rangle$$

Let $\theta^{(t)}$ denote the parameter value at the t-th update in gradient-based training. For any $i = 1 \ldots d$, we define $\theta_{-i}^{(t)}$ to be equal to $\theta^{(t)}$ except that its i-th element is zero. The approximation of $J_k(\theta)$ around $a = \theta_{-i}^{(t)}$ at $\theta = \theta^{(t)}$ is

$$\begin{aligned} J_k(\theta^{(t)}) &\approx J_k(\theta_{-i}^{(t)}) + \left\langle \nabla J_k(\theta_{-i}^{(t)}), \theta^{(t)} - \theta_{-i}^{(t)} \right\rangle \\ &= J_k(\theta_{-i}^{(t)}) + \frac{\partial J_k(\theta^{(t)})}{\partial \theta_i} \times \theta_i^{(t)} \end{aligned}$$

Rearranging and taking the absolute value, we have

$$\sigma_{i,k}^{(t)} = \left| \frac{\partial J_k(\theta^{(t)})}{\partial \theta_i} \times \theta_i^{(t)} \right| \approx \left| J_k(\theta^{(t)}) - J_k(\theta_{-i}^{(t)}) \right| \tag{4.1}$$

which is easily computable and can be viewed as measuring how sensitive the i-th parameter is with respect to the k-th task in the t-th iteration of training. We propose to use this quantity, previously used in the model pruning literature [87], to encourage task specialization during training. We define a conditional distribution over K tasks by

$$q(k|\theta^{(t)}, t, i) = \frac{\exp(\bar{\sigma}_{i,k}^{(t)}/\tau_t)}{\sum_{k=1}^{K} \exp(\bar{\sigma}_{i,k}^{(t)}/\tau_t)} \tag{4.2}$$

where $\tau_t > 0$ is a temperature and $\bar{\sigma}_{i,k}^{(t)}$ is an appropriately normalized and amortized estimation of $\sigma_{i,k}^{(t)}$ in Eq. (4.1) (see Section 4.3.2.1). Assuming training examples are sampled to roughly balance the size across tasks (i.e., $N_k \approx N_{k'}$), we take the following gradient step for the i-th parameter in the t-th iteration:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \eta \sum_{k=1}^{K} q(k|\theta^{(t)}, t, i) \times \frac{\partial J_k(\theta^{(t)})}{\partial \theta_i}$$

Note that this is a per-parameter adaptive learning. Each parameter $\theta_i \in \mathbb{R}$ maintains a distribution over K tasks and is updated more aggresively for tasks that θ_i is sensitive to.

4.3.2.1 Sensitivity normalization

The d parameters $\theta^{(t)}$ can be of very different magnitudes. To reduce the parameter-wise variance in the sensitivity scores, for task k we divide the scores by the median of across all parameters with respect to task k:

$$\tilde{\sigma}_{i,k}^{(t)} = \frac{\sigma_{i,k}^{(t)}}{\mathrm{median}_{j=1 \dots d}(\sigma_{j,k}^{(t)})}$$

We use the median instead of the mean to account for the long tail distribution of task-specific sensitivity scores. We also use momentum to amortize the scores: assuming some $\beta > 0$

$$\bar{\sigma}_{i,k}^{(t)} = (1-\beta)\bar{\sigma}_{i,k}^{(t-1)} + \beta\tilde{\sigma}_{i,k}^{(t)}$$

where $\bar{\sigma}_{i,k}^{(0)} = 0$. This is the final version of sensitivity that we use in Eq. (4.2). The algorithm in matrix form is given in Algorithm 1 (Appendix B.1).

4.4 Experiments

4.4.1 Setup

Datasets. We follow [81] and use eight tasks from KILT [6] for training and evaluation. We randomly downsample the training data of the two largest datasets (T-REx and zsRE) to the same order of magnitude as the rest. All the datasets share the same knowledge base of 36 million disjoint 100-token Wikipedia passages preprocessed by Maillard *et al.* [81]. The data statistics and other data-related details can be found in Appendix B.2.

Evaluation. We use the page-level R-precision (the suggested main metric in KILT) to measure the retrieval performance. Page-level R-precision is the fraction of the R gold pages captured by the retriever in the top-R candidates. We map the retrieved passages to the their corresponding pages and use official KILT evaluation scripts to evaluate the page-level R-precision. We also report passage-level R-precision proposed by Maillard *et al.* [81] on dev sets in Appendix B.5. We use TREC Eval[2] to evaluate the passage-level R-precision.

Model details. We initialize our dual encoder with the official T5-base [2] checkpoint. The query encoder and passage encoder share weights. Following the ANCE [82] training paradigm, we first warmup our model for 20 epochs with BM25 hard negatives by naive multitask learning with task prefix. Then we train the model for 8 ANCE episodes with the model-mined hard negatives refreshed at the begining of each ANCE episode. We adopt naive multitask learning with task prefix for the first 7 ANCE episodes and apply the adaptive learning introduced in Section 4.3.2 for the last ANCE episode to improve the performance further. We use Adam [96] with a linear learning rate decay schedule with warmup proportion 0.1 over 3 epochs for each ANCE iteration. We provide more details and hyperparameters in Appendix B.3.

4.4.2 Main Results

We refer to our model as **TACO**, which stands for **TA**sk spe**C**ialty **O**ptimization. Table 4.1 and Table 4.2 show our main results on the KILT validation data and test data respectively. Fewer comparable baselines are available for KILT test data than for KILT validation data.

Let avg val denote average validation page-level R-Precision. TACO achieves the best performance on 4 out of 8 tasks for both validation and test data. The performance is either the second best or close to the second best except AIDA, an entity linking dataset favoring autoregressive retrieval models over dense retrieval models [22]. TACO outperforms the previous multitask dense retrieval model MT-DPR [81] significantly (+7.34% avg val). TACO also achieves better performance compared with current top performing multitask autoregressive retrieval models in comparable setting (finetuned purely on KILT). TACO outperforms BART$_{mt}$ (+3.33% avg val) with smaller model size (T5-base vs Bart-large). Compared with BART$_{mt}$, CorpusBrain$_{mt}$ employs additional pretraining and yields significant improvement over BART$_{mt}$ (+2.15% avg val). TACO still outperforms CorpusBrain$_{mt}$ (+1.18% avg val) with smaller model size and no additional pre-

[2]https://trec.nist.gov/trec_eval/

	Fact Check.	Ent. L.	Slot Filling		Open Domain QA			Dial.	
Model	**FEV**	**AY2**	**T-REx**	**zsRE**	**NQ**	**HoPo**	**TQA**	**WoW**	**Avg**
				Baselines.					
BM25*	50.13	3.47	58.60	66.43	25.83	43.95	29.44	27.50	38.17
$\text{BART}^{\dagger}_{mt}$	81.92	89.17	75.18	91.08	58.62	48.69	67.64	50.98	70.41
$\text{CorpusBrain}^{\dagger}_{mt}$	<u>82.06</u>	**90.84**	<u>77.62</u>	98.26	59.10	50.07	68.78	<u>53.75</u>	<u>72.56</u>
MT-DPR*	74.72	83.78	69.18	77.23	61.51	44.21	61.95	39.70	64.04
Task-specific DPR*	73.60	81.77	69.08	97.74	<u>63.24</u>	46.63	65.12	40.32	67.19
Task-specific BART†	80.03	87.98	74.46	93.91	50.96	39.21	66.13	50.75	67.93
Task-specific CorpusBrain†	81.77	<u>90.36</u>	76.90	<u>98.49</u>	57.67	**50.62**	<u>69.25</u>	53.60	72.33
Task-specific (ours)	74.28	85.28	77.18	**99.38**	**65.39**	46.79	69.08	53.63	71.38
				Non-Comparable Models (For Reference).					
$\text{CorpusBrain}^{\dagger}_{mt+BLINK}$	85.03	92.86	80.22	98.49	64.61	52.23	71.71	59.72	75.61
GENRE†	84.68	92.75	79.68	94.84	64.26	51.82	71.11	56.32	74.43
TABi [97]	85.8	-	82.0	95.2	62.4	52.7	71.5	51.8	-
TACO	**86.17**	84.64	**78.12**	97.91	61.86	<u>50.61</u>	**69.62**	**60.97**	**73.74**

Table 4.1: Page-level R-precision on KILT validation data. **Bold** indicates the best model and <u>underline</u> indicates the second. † and $*$ mark results from Chen *et al.* [92] and Maillard *et al.* [81] respectively. The non-comparable models are trained on additional data or use extra information. We list them only for reference not for comparison. Taks-specific models use a separate retriever for each task while all the other models use a single retriever across all the tasks.

training. We also list various top performing multitask retrieval models for reference but not for comparison because they are not in comparable setting. Both GENRE and $\text{CorpusBrain}_{mt+BLINK}$ are finetuned on a large amount of additional training data besides KILT training data. Specifically, they also use BLINK training data [67] for finetuning, which contains 8.9M annotated wikipedia sentences. TABi [97] uses extra type labels information and leverages knowledge graph that is very effective for retrieval. TACO even rivals these non-comparable models on all the tasks except AIDA.

TACO is the only model that outperforms strong task-specific models noticeably. Our task-specific baseline is significantly stronger than the task-specific DPR, likely due to better training paradigm (ANCE) and better model (T5 vs BERT). Task-specific CorpusBrain is even stronger, especially for FEVER and AIDA. Only TACO and CorpusBrain_{mt} outperform the strong task-specific models. TACO achieves a 2.36% improvement over its task-specific counterpart and a

	Fact Check.	Ent. L.	Slot Filling		Open Domain QA			Dial.	
Model	**FEV**	**AY2**	**T-REx**	**zsRE**	**NQ**	**HoPo**	**TQA**	**WoW**	**Avg**
				Baselines.					
TF-IDF†	50.9	3.7	44.7	60.8	28.1	34.1	46.4	49.0	39.7
SEAL‡	<u>81.4</u>	-	62.1	91.6	**63.2**	**58.8**	68.4	<u>57.5</u>	-
MT-DPR*	74.5	26.5	69.5	80.9	59.4	42.9	61.5	41.1	57.0
MT-DPR$^{‡}_{WEB}$	74.8	-	75.6	89.7	59.8	45.4	58.9	41.5	-
Task-specific (ours)	73.22	<u>79.52</u>	<u>77.00</u>	**99.15**	<u>60.87</u>	46.50	**69.12**	55.03	70.05
				Non-Comparable Models (For Reference).					
CorpusBrain$^{†}_{mt+BLINK}$	84.07	89.98	79.98	98.27	60.32	51.80	70.19	64.79	74.93
GENRE†	83.64	89.85	79.42	95.81	60.25	51.27	69.16	62.88	74.04
TABi [97]	84.4	-	81.9	96.2	62.6	53.1	70.4	59.1	-
TACO	**84.07**	**80.64**	**77.22**	<u>98.21</u>	60.80	<u>50.70</u>	<u>68.45</u>	**62.64**	**72.84**

Table 4.2: Page-level R-precision on KILT test data. **Bold** indicates the best model and <u>underline</u> indicates the second. †, $*$ and ‡ mark results from Chen *et al.* [92], Maillard *et al.* [81] and Bevilacqua *et al.* [25] respectively. The non-comparable models are trained on additional data or use extra information. We list them only for reference not for comparison.

	Fact Check.	Ent. L.	Slot Filling		Open Domain QA			Dial.	
Variants	**FEV**	**AY2**	**T-REx**	**zsRE**	**NQ**	**HoPo**	**TQA**	**WoW**	**Avg**
TACO	**86.17**	84.64	**78.12**	**97.91**	61.86	50.61	**69.62**	**60.97**	**73.74**
w/o task prefix	85.71	84.68	74.82	94.68	61.05	49.38	67.79	58.81	72.12
w/o adaptive	84.81	85.49	75.00	92.24	62.81	51.47	68.95	60.54	72.66
w/o task prefix w/o adaptive	84.03	85.62	70.96	86.04	62.46	49.78	66.04	59.95	70.61
task query encoder	82.71	**87.56**	72.72	85.15	**64.01**	49.74	69.12	55.93	70.87
task type marker	84.49	85.51	73.88	89.37	62.85	50.97	67.70	60.02	71.85
PCG [83]	84.97	85.26	74.90	91.43	62.67	51.47	68.54	60.48	72.47
CGD [85]	82.25	80.39	71.62	83.40	62.67	49.66	66.73	59.33	69.51
GradNorm [86]	84.70	85.28	75.32	91.73	63.80	**51.97**	69.30	60.31	72.80

Table 4.3: Ablation study results on KILT validation data. We report page-level R-precision. **Bold** indicates the best variant. Each line makes a single or multiple changes from the TACO model. The performance of the recent general multitask algorithms, PCG [83], CGD [85] and GradNorm [86], are obtained from our own implementation.

1.41% improvement over the task-specific CorpusBrain, but CorpusBrain$_{mt}$ is only slightly better than its task-specific counterpart (+0.23% avg val).

4.4.3 Analysis

4.4.3.1 Ablation Study

Table 4.3 shows the results of ablation studies on KILT validation data.

Model components. We first conduct experiments to understand the impact of individual components of our model. Removing task prefix results in 1.62% R-precision decrease and disabling adaptive learning yields 1.08% R-precision decrease. Removing both task prefix and adaptive learning significantly degrades the performance (-3.13%). This demonstrates that both task prefix and adaptive learning contribute to the effectiveness of TACO.

Query variants. We conduct experiments to investigate other query side variants besides task prefix. These variants are not trained with adaptive learning and only change the query input format or model. Leveraging task-specific query encoder yields slightly better performance (70.87% vs 70.61%), but is outperformed by task prefix significantly (70.87% vs 72.66%). The task type marker introduced in Maillard *et al.* [81] is not helpful for BERT-based MT-DPR, but we find them effective for our T5-based model. This is likely because T5 is pretrained for multitasking. We conduct experiments to leverage their task type markers for our model. Using task type markers (i.e., 5 symbols indicating the 5 classes of task in KILT) leads to 1.24% R-precision improvement (71.85% vs 70.61%), but is less effective than our fine-grained dataset-level task prefix (71.85% vs 72.66%).

Mutltitask learning variants. We compare our adaptive learning method with recent general multitask learning algorithms with our own implementation. PCG [83] focuses on mitigating the conflict of gradients from different tasks. It performs on par with the "w/o adaptive" variant (72.47% vs 72.66%), but underperforms TACO which leverages our adaptive learning (72.47% vs 73.74%). This shows that the gradient conflict is not the main bottleneck in our multitask retrieval setting. CGD [85] aims to improve multitask learning by encouraging update towards common directions of different tasks, which is opposite to our method that encourages task specialties. It performs much worse than TACO (69.51% vs 73.74% and lags behind the "w/o adaptive" variant significantly (69.51% vs 72.66%). This shows that we should encourage task specialty rather than emphasizing tasks shared part for multitask retrieval. GradNorm [86] tries to weight different tasks losses by using the average gradient norm. It performs slightly better than the naive "w/o adaptive" variant (72.47% vs 72.66%). Our adaptive learning method achieves descent improvement over GradNorm (73.74% vs 72.80%). Note that our adaptive update is more fine-grained and critically different because we adjust learning rates along both task dimension and parameter dimension compared with GradNorm that only do loss re-weighting.

Adaptive learning. We consider variations of the main version of adaptive learning which is applied only in the last ANCE episode. Specifically, we investigate the impact of applying adaptive learning to the last four ANCE episodes using an exponential softmax temperature decay scheduler. This approach yields an average page-level R-precision of 73.47%. In comparison, when adaptive learning is applied only to the last ANCE episode, we achieve an average page-level R-precision of 73.74%. These results suggest that extending adaptive learning to more ANCE episodes does not yield improvement. Additionally, we examine the effectiveness of encouraging task specialization within adaptive learning. For this purpose, we focus on the second ANCE episode and experiment with positive softmax temperature (encouraging task specialty) and negative softmax temperature (discouraging task specialty). Encouraging task specialization results in an average page-level R-precision of 70.53%, while discouraging task specialization leads to an average page-level R-precision of 68.39%. In comparison, the performance of the standard multitask baseline at the second ANCE episode is 69.28%. These results highlight the benefits of encouraging task specialization and the detrimental effect of discouraging task specialization within adaptive learning. Normalizing task sensitivity using the median is preferred over using the mean or not applying any normalization, as different tasks exhibit variations in magnitude while sharing similar distribution shapes (see Figure 4.2).

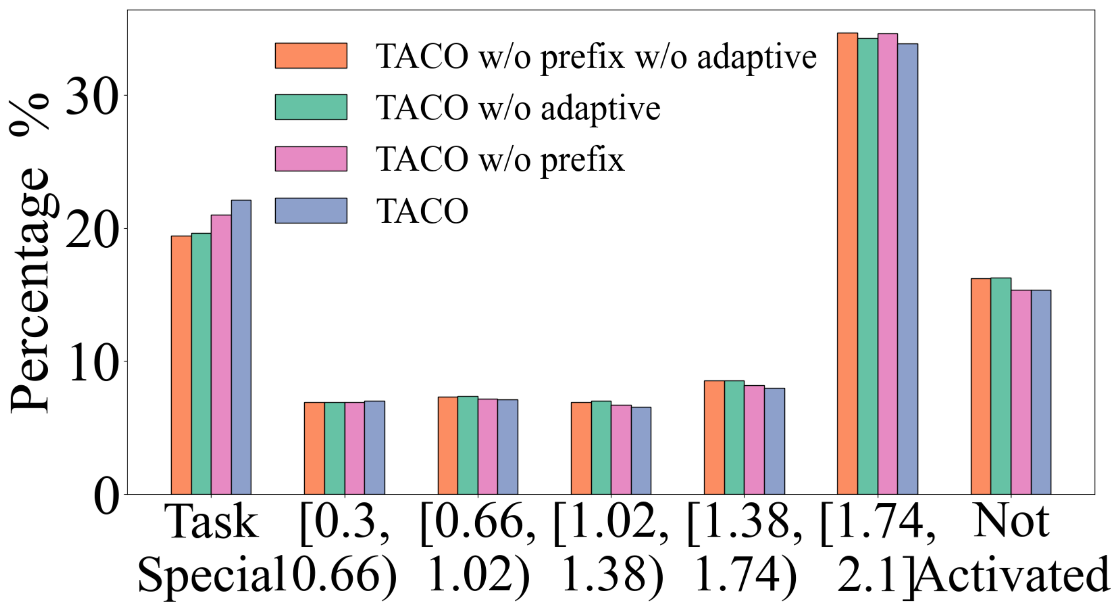
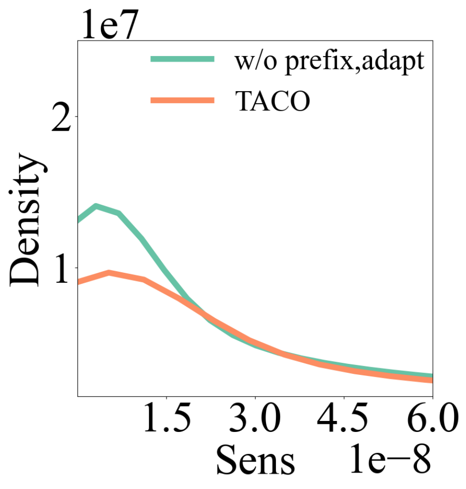


Figure 4.1: Task entropy histograms for model variants

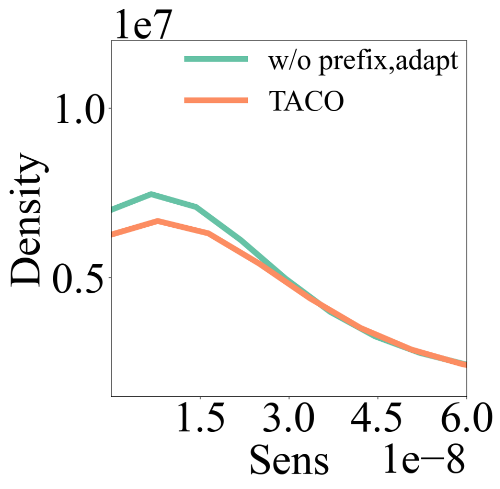
4.4.3.2 Task Specialization

Figure 4.1 plots the histograms of task entropy for the learned parameters. The task entropy for each parameter is calculated with the distribution defined in equation 4.2. We first group parameters into two special bins. The first is a “Task Specific” bin that includes parameters whose entropy is smaller than 0.3, which is the entropy of 95% probability on one task and the 5% uniformly on the rest seven. The “Not Activated” bin includes parameters whose sensitivity w.r.t. all tasks is near zero ($< 1e-8$). TACO significantly improves the fraction of task specific parameters to 22%, in comparison with 19% in naive multitask model (w/o prefix w/o adaptive). It also reduces the fraction of not activated parameters, showing optimizing task specialty also better utilizes the model capacity.

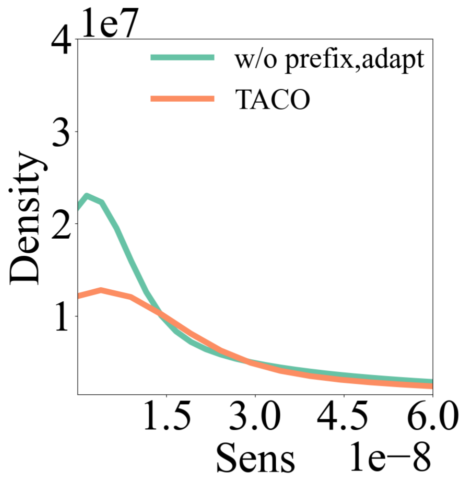
Figure 4.2 plots the kernel density estimated distribution of task-specific sensitivity in TACO and the standard multitask model for four KILT tasks. We drop outliers that deviates significantly from the median to ease visualization. Notably, TACO exhibits a noticeable reduction in the peak on the low sensitivity side for each task compared to the standard multitasking model. This obser-



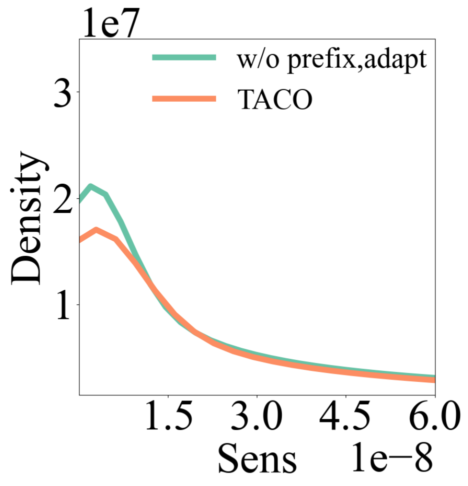
(a) NQ



(b) WoW



(c) T-REx



(d) FEV

Figure 4.2: Task-specific sensitivity density distribution on the training data of four KILT tasks. The final models are used. The x-axis is sensitivity, and we drop outliers that are far from the median to ease visualization.

vation suggests that TACO activates a larger number of parameters and enhances their sensitivity towards individual tasks.

4.4.3.3 Additional Benchmark

To test the performance of TACO in a different setup other than KILT, we constructed an additional benchmark containing MS-MARCO [98], ZESHEL [66], a document-level version of

	MS	**ZES**	**FEV**	**NQ**	**Avg**
Task-specific	73.3	67.3	90.0	71.8	75.6
TACO	85.8	67.6	91.2	76.8	80.4
w/o adapt	85.9	67.5	91.3	76.6	80.3
w/o prefix, adapt	86.2	68.1	92.2	76.4	80.7
PCG	86.1	67.9	91.7	76.9	80.7
CGD	86.8	69.1	94.4	76.2	81.6
GradNorm	86.0	67.3	91.7	76.9	80.5

Table 4.4: Recall@100 on an additional benchmark containing MS-MARCO (MS), ZESHEL (ZES), FEVER (FEV), and Natural Questions (NQ).

FEVER from BEIR [99], and Natural Questions from KILT. We chose this combination for a few reasons. First, we found that few public datasets outside KILT provide sufficiently large and high-quality training data other than MS-MARCO and ZESHEL. Second, each task now has its own KB to retrieve from, making this a rather different setup from KILT in which all tasks share one KB. We compare task-specific retrievers and multitask retrievers trained by TACO and other methods. Table 4.4 shows their recall at 100 on the validation split. We see that multitasking is clearly beneficial for this benchmark. The best performance is obtained by CGD and it is the only multitask optimization method that yields noticeable improvements over the standard multitask model. Given that CGD aims to improve multitask learning by encouraging update towards common directions of different tasks, we hypothesize that the need for task specialization is diminished here because the tasks are more similar in difficulty (e.g., in KILT, T-REx and zsRE are much easier than HotpotQA). This experiment sheds light on what multitask settings most benefit from task specialization.

4.5 Conclusions

Multitask retrieval has compelling practical advantages such as model simplicity and memory efficiency, but it lags behind task-specific retrieval in the existing literature. We have shown that it is possible to significantly improve the performance of multitask retrieval by promoting task specialization. The key steps are the use of a base model optimized for multitasking with appropriate prompting and a per-parameter adaptive learning technique that upweights the task gradients by the parameters' sensitivity to the task losses. We have achieved strong results on the KILT retrieval benchmark.

CHAPTER 5
RETRIEVAL-AUGMENTED GENERATION WITH IMPLICIT QUERIES

This chapter is adapted from joint work with Xi Victoria Lin, Karl Stratos, Wen-tau Yih and Mingda Chen entitled “ ImpRAG: Retrieval-Augmented Generation with Implicit Queries” [100].

Retrieval-Augmented Generation (RAG) systems traditionally treat retrieval and generation as separate processes, requiring explicit textual queries to connect them. This separation can limit the ability of models to generalize across diverse tasks. In this chapter, we propose a query-free RAG system, named ImpRAG, which integrates retrieval and generation into a unified model. ImpRAG allows models to implicitly express their information needs, eliminating the need for human-specified queries. By dividing pretrained decoder-only language models into specialized layer groups, ImpRAG optimizes retrieval and generation tasks simultaneously. Our approach employs a two-stage inference process, using the same model parameters and forward pass for both retrieval and generation, thereby minimizing the disparity between retrievers and language models. Experiments on 8 knowledge-intensive tasks demonstrate that ImpRAG significantly enhances both retrieval and generation performance, with exact match scores increasing by 3.6-11.5 points and retrieval recalls improving by 5.0-23.2 points for unseen tasks with diverse formats, highlighting its effectiveness in enabling models to articulate their own information needs and generalize across tasks. Our analysis underscores the importance of balancing retrieval and generation parameters and leveraging generation perplexities as retrieval training objectives for enhanced performance.

5.1 Introduction

Retrieval-Augmented Generation (RAG; [33, 7, 40]) typically involves two key operations: retrieval and generation. RAG systems retrieve relevant information to enhance generation models, enabling them to respond more effectively to prompts by providing long-tail knowledge or up-

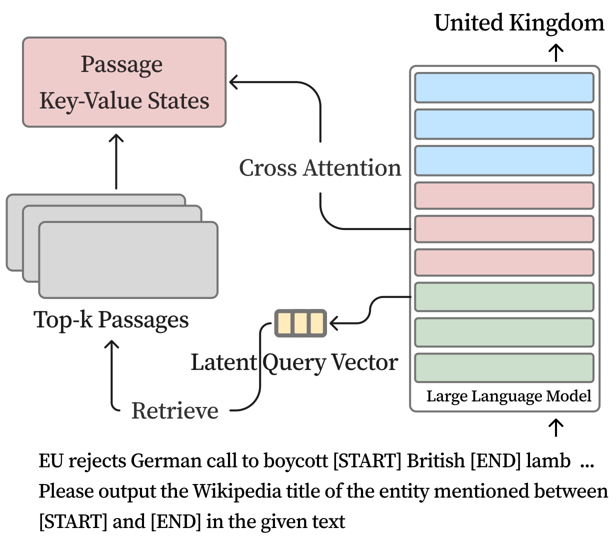

United Kingdom
Passage Key-Value States
Cross Attention
Top-k Passages
Latent Query Vector
Retrieve
Large Language Model
EU rejects German call to boycott [START] British [END] lamb ...
Please output the Wikipedia title of the entity mentioned between
[START] and [END] in the given text

Figure 5.1: Diagram illustrating the inference process of ImpRAG on the entity linking task. We divide decoder-only LLMs into three layer groups for specialized finetuning: bottom (green), middle (red), and top (blue). The bottom layers are optimized for retrieval tasks. The middle and top layers handle the reading of retrieved passages, with cross-attention disabled in the top layers to reduce memory consumption. Standard RAG systems would require a task-specific design of queries (e.g., use the substring "British" as the query in the shown example). In contrast, ImpRAG uses implicit queries, eliminating the need for explicit specification of queries and allowing models to generalize across unseen tasks with varied formats.

to-date information. While effective, traditional approaches often treat retrieval and generation as separate processes, connected by queries.[1] Consequently, these approaches usually require explicit specification of textual queries. By definition, queries express one's uncertainties; however, in RAG systems, instead of models expressing their information needs, humans must do this for them. This separation can lead to a disconnect between what large language models (LLMs) require and what retrievers assume is necessary. More importantly, it restricts the models' ability to generalize across diverse, unseen tasks during testing. Therefore, in this chapter, we explore the development of a *query-free* RAG system, enabling models to articulate their own information needs without additional human intervention.

To achieve this, we introduce ImpRAG, a novel approach that integrates retrieval and generation into a unified model and process. This allows models to convey their own information needs

[1]In this chapter, we use the term "queries" to refer to textual queries used in an information retrieval setup, unless otherwise specified. This is distinct from queries in the context of self-attention within Transformer architectures.

implicitly, reducing the need for prior knowledge of test tasks and for humans to formulate explicit textual queries in advance. At its core, ImpRAG aims to enable retrieval capabilities through retrieval heads in self-attention. Building upon pretrained decoder-only language models, ImpRAG divides the layers into three groups: the bottom group for retrieval and the middle and top groups for reading and generation.

Figure 5.1 illustrates an example of applying ImpRAG to the entity linking task, where models are tasked with linking the mention ”British” to an entity in Wikipedia, given the context paragraph. A typical RAG model would require the design of a separate query template, such as using only the mention text, to achieve reasonable retrieval performance. In contrast, ImpRAG uses implicit queries and can perform retrieval and generation jointly without the need for additional template design, making it more generalizable.

During training, we optimize two objectives simultaneously: generation loss and retrieval loss. The generation loss is the standard causal language modeling loss, while the retrieval loss first utilizes pseudo labels generated by trained retrievers to warm up the retrieval ability and then self-improves using its own generation log likelihood for the remainder of the training.

At inference time, we employ a two-stage process. First, we embed passages using the bottom layer for retrieval, and then utilize the top layer group to read the retrieved passages and generate the final responses. By leveraging the same forward pass and model parameters for both retrieval and generation, ImpRAG reduces the disparity between retrievers and LLMs.

In experiments, we train models on datasets that either require retrieval or do not. The datasets requiring retrieval are used to enhance retrieval performance, while those not requiring retrieval are used to improve models’ instruction-following capabilities. We evaluate the models on 8 knowledge-intensive tasks, focusing on different aspects: basic question answering, multihop reasoning, and instruction following. We also establish strong baselines that perform RAG in the retrieve-then-generate paradigm, including RA-DIT [38], a method that iteratively updates LLMs and retrievers to better align the two.

Our experiments demonstrate that ImpRAG achieves slightly better performance on 4 tasks with formats similar to the training tasks, with an improvement of 0.2-0.6 points in exact match scores, all without the need for additional model parameters. Moreover, it significantly outperforms previous approaches on unseen test tasks with more diverse formats, achieving improvements of 3.6-11.5 points in exact match scores and 5.0-23.2 points in retrieval recalls. This highlights the effectiveness of enabling models to articulate their own information needs. Our analysis indicates that carefully selecting layer group boundaries that balance the parameters used for retrieval and generation, using both trained retrievers for warmup and then self-improve by leveraging generation perplexities as retrieval training objectives, and instruction tuning training datasets is crucial for achieving superior performance in ImpRAG. Our analysis also reveals that ImpRAG is effective in transferring supervision from generation tasks to retrieval tasks, showing the potential of using an unified model architecture for performing retrieval and generation jointly.

5.2 Related Work

There has been a lot of work on using the retrieve-then-generate paradigm for RAG [7, 40, *inter alia*]. Many efforts in this line of work have focused on optimizing retrievers using training signals from generation models, and optionally, the reverse [33, 7, 39, 40]. Although the specifics can differ, these approaches generally utilize distinct models and input templates for the retrieval and generation phases. A closely related study is that of Jiang *et al.* [101], which seek to use the same model for retrieval and generation. However, their research primarily focuses on encoder-decoder style models and their models still rely on separate input templates for retrieval and generation. Another related work by Zhang *et al.* [102] explores the use of special tokens for retrieval, but their study emphasizes in-domain task performance rather than unseen task generalization.

This chapter is also related to research on query formulation in the context of multihop question answering, where previous studies typically generate textual queries by prompting LLMs, followed by retrieval using a separate retriever [103, 43, 42, 104, 105, *inter alia*]. Chen *et al.* [106] enable LLMs to generate textual queries through synthetic data generation. Additionally, this chapter is connected to memory architectures in RAG [107, 108], which aim to utilize the key-value (KV) caches of LLMs to reduce computational costs, rather than focusing on minimizing the disparities between generation and retrieval.

Another relevant area of research is instruction tuning for RAG. Lin *et al.* [38] perform instruction tuning for both retrievers and LLMs and then align them through iterative updates. Wang *et al.* [109] conduct instruction tuning for RETRO-like models [110, 111]. Zhang *et al.* [112] align retrievers with LLMs using synthetic data generated by LLMs. Unlike our work, these studies still treat retrieval and generation as separate processes. In a similar vein, researchers have tried to teach retrievers to follow instructions for building general-purpose information retrieval systems [113, 114, 115, 116]. Since ImpRAG enables its retrieval capabilities by using self-attention, it is related to research on investigating retrieval heads in the context of long context LLMs [117].

5.3 Method

We build on an autoregressive pretrained language model and enable it to perform retrieval and generation jointly. Our model, **ImpRAG**, is based on the LLaMA 3 family [5], with architectural modifications to support retrieval and retrieval-augmented generation. At a high level, the layer grouping strategy of ImpRAG is inspired by the observation that LLMs learn distinct functions at different layers [118]. Consequently, we have designed the layer groups to align with the capabilities required for retrieval-augmented generation, i.e., retrieval and generation.

5.3.1 Architecture

Layer Slicing. We partition an N-layer language model vertically into three groups, as illustrated in Figure 5.1. The bottom group, spanning layers 0 to b, is denoted as $\mathcal{L}_{\mathcal{B}}$. The middle group, from layer b to t, is denoted as $\mathcal{L}_{\mathcal{M}}$, and the top group, from layer $t+1$ to $N-1$, as $\mathcal{L}_{\mathcal{T}}$. Note that $\mathcal{L}_{\mathcal{B}}$ and $\mathcal{L}_{\mathcal{M}}$ share layer b, while $\mathcal{L}_{\mathcal{M}}$ and $\mathcal{L}_{\mathcal{T}}$ are disjoint. The layer boundaries b and t are treated as hyperparameters and can be tuned to optimize performance across different model configurations.

Bottom Layers as Retriever. We repurpose the bottom group $\mathcal{L}_{\mathcal{B}}$ to act as a *retriever*, in addition to its standard decoder functionality. Specifically, we apply pooling last-token pooling over the attention query or key states at the final layer b in $\mathcal{L}_{\mathcal{B}}$. Unlike prior work [119], we retain the original causal attention in the bottom layers rather than enabling bidirectional attention, as we do not observe any performance improvement from this modification.

Let h_k be the number of key attention heads, g the number of query attention groups (as in Grouped-Query Attention [120]), and d_h the head dimension. For a query input, we apply last-token pooling by taking the query attention state of its final token, resulting in a grouped query embedding $\mathbf{E}_q^g \in \mathbb{R}^{(h_k g) d_h}$. We then average the attention heads within each group to obtain the final query embedding $\mathbf{E}_q \in \mathbb{R}^{h_k d_h}$.[2] Similarly, for each corpus passage, we extract the key attention state of its last token to compute the passage embedding $\mathbf{E}_p \in \mathbb{R}^{h_k d_h}$. Similarity between query and passage embeddings is computed via dot product:

$$s(q,p) = \mathbf{E}_q \cdot \mathbf{E}_p \tag{5.1}$$

We choose to pool over query and key attention states based on the intuition that their dot product underlies the attention mechanism and is pretrained to capture token-wise relevance. By aggregating these signals across tokens, we aim to capture query-passage-level semantic relevance.

Middle Layers as Reader. The middle layer group $\mathcal{L}_{\mathcal{M}}$ functions as a *reader* by enabling cross-attention from the input query tokens to the retrieved passage tokens, thereby incorporating external information into the query representation. Given k retrieved passages, we jointly encode the concatenation of all k passages to form the key and value states for layers b through t. Cross-attention is then performed from the query's attention states to these key and value states, allowing the model to read and integrate relevant content from the passages. This aligns with prior findings that middle layers of language models are particularly effective at attending to and integrating long-range contextual information [121, 107].

[2]Our preliminary results show that taking average heads works slightly better than using individual heads.

Top Layers Disable Cross-Attention. In the top layer group $\mathcal{L}_T$, we optionally disable cross-attention from the input query tokens to the retrieved passage tokens solely to reduce computational and memory overhead. This design choice is made for efficiency purposes; empirically, we find it results in only a minor performance drop when the layer boundary t is properly tuned as a hyperparameter.

Position IDs. Language models using RoPE [122] are highly sensitive to position IDs. To prevent interference between the query and passage position encodings during reading, we shift the query's position IDs to the right rather than starting from zero. Let l_{max} denote the maximum passage length and k the number of retrieved passages. We shift the query position IDs by $k \cdot l_{max}$ tokens to account for the total length.

5.3.2 Training

We train ImpRAG using a multi-task objective that jointly optimizes generation and retrieval:

$$J = J_{\text{gen}}(r \mid q, \mathcal{C}) + \lambda \cdot J_{\text{ret}}(q, \mathcal{C}) \tag{5.2}$$

Here, $J_{\text{gen}}(r \mid q, \mathcal{C})$ denotes the generation loss, implemented as the standard causal language modeling loss over the response tokens r, conditioned on the input query q and a set of sampled candidate passages $\mathcal{C}$. The term $J_{\text{ret}}(q, \mathcal{C})$ denotes the retrieval loss, computed over the query q and the same set of candidate passages $\mathcal{C}$, and is further detailed in the two-stage formulation described in Section 5.3.2.1. The hyperparameter λ balances the relative importance of the retrieval loss, allowing us to control the trade-off between retrieval accuracy and generation quality during training.

5.3.2.1 Retrieval Objective

While the overall training objective remains consistent across both stages—combining generation and retrieval losses as in (5.2)—the retrieval loss component J_{ret} varies depending on the training phase. In this section, we describe the two-stage training process used to endow ImpRAG with strong retrieval capabilities.

Warmup. Since the pretrained language model is not inherently optimized for retrieval, we begin with a warmup stage that introduces basic retrieval ability. We adopt a Multi-Label NCE loss [123] as the retrieval objective and construct supervision using pseudo-labeled data generated by a strong off-the-shelf retriever, Contriever-MSMARCO [124]. For each query q, we retrieve the top-5 passages as pseudo-positive examples, denoted by $\mathcal{P}(q)$. We then sample a small set of pseudo hard negatives, denoted by $\mathcal{N}_h(q)$ (e.g., $|\mathcal{N}_h(q)| < 10$), from passages ranked 10–50.[3] While these passages may still be somewhat relevant, they are less likely to contain the key information necessary to answer the query. This selection introduces meaningful retrieval difficulty. We also use in-batch negatives across devices as additional random negatives $\mathcal{N}_r(q)$. The full negative set is $\mathcal{N}(q) = \mathcal{N}_h(q) \cup \mathcal{N}_r(q)$, and the candidate set is $\mathcal{C} = \mathcal{P}(q) \cup \mathcal{N}(q)$. The retrieval loss for this stage is defined as:

$$J_{\text{ret}}(q, \mathcal{C}) = -\sum_{p \in \mathcal{P}(q)} \log \left(\frac{\exp(s(q,p))}{\exp(s(q,p)) + \sum_{p' \in \mathcal{N}(q)} \exp(s(q,p'))} \right) \tag{5.3}$$

Self-Distillation. To further enhance retrieval performance, we employ language model perplexity distillation [39], which assesses how much each candidate passage improves the language model's likelihood of generating the ground-truth response, conditioned on the query. Specifically, for each candidate passage $p \in \mathcal{C}$, we compute the log-likelihood of the gold response r given the concatenation of p and q, denoted as $\log P_{\text{LM}}(r \mid p, q)$. This defines a soft target distribution over

[3]We find this approach effective in preliminary experiments, though we did not perform extensive hyperparameter tuning.

candidate passages:

$$P_T(p \mid q, r) = \frac{\exp(\log P_{\text{LM}}(r \mid p, q))}{\sum_{p' \in \mathcal{C}} \exp(\log P_{\text{LM}}(r \mid p', q))} \tag{5.4}$$

We also define the retrieval model's predicted distribution based on the similarity scores:

$$P_R(p \mid q) = \frac{\exp(s(q, p))}{\sum_{p' \in \mathcal{C}} \exp(s(q, p'))} \tag{5.5}$$

The retrieval loss is then computed as the KL divergence between the target and predicted distributions:

$$J_{\text{ret}}(q, \mathcal{C}) = \text{KL}\left(\overline{P_T(p \mid q, r)} \parallel P_R(p \mid q)\right) \tag{5.6}$$

Here, $\overline{P_T(p \mid q, r)}$ indicates that gradients are not backpropagated through the target distribution. Note that this stage also involves joint training; the only difference from the warmup phase lies in the retrieval loss J_{ret}.

5.3.3 Inference

At inference time, we first embed all passages in the knowledge corpus using the bottom layer group $\mathcal{L}_B$ of the model. These embeddings are stored in an approximate nearest neighbor (ANN) index (e.g., FAISS [19]) hosted on a remote server for efficient retrieval.

As illustrated in Figure 5.1, given a query, the ImpRAG model performs the following steps to generate a response:

1. The bottom layers $\mathcal{L}_B$ encode the input query and generate a query embedding, which is sent to the remote ANN search server.

2. The ANN server retrieves the top-k most relevant passages based on the query embedding and returns their passage IDs.
3. The middle layers $\mathcal{L}_M$ continue processing the information by applying cross-attention to the KV states of the retrieved passages.
4. The top layers $\mathcal{L}_T$ complete the encoding and decoding process without cross-attention, generating the next token.
5. The above steps are repeated at each decoding step. Notably, the query embeddings are computed only once at the end of the input prompt, and passage retrieval is not re-triggered thereafter.[4] In subsequent decoding steps, cross-attention continues to use the cached key-value states, and this process repeats until the model reaches a stopping criterion (e.g., an end-of-sequence token).

5.4 Experiment

5.4.1 Experimental Setup

Training. For training, we consider two types of datasets: (1) datasets requiring retrieval knowledge: NaturalQuestions (NQ; [125]) and HotpotQA (Hopo; [126]); and (2) datasets without requiring retrieval knowledge, where we use the instruction tuning datasets from Lin *et al.* [38] (see Appendix C.4 for a complete list of these datasets). Inspired by Chen *et al.* [127], we also incorporate two synthetic, retrieval-free tasks into the training to enhance instruction-following capabilities: phrase denoising, and next/previous sentence generation. The training data for phrase denoising is generated by prompting LLMs (we use Llama-3.1 70B) with a paragraph from Wikipedia. For the sentence generation task, we construct it randomly using content from Wikipedia.

For all these datasets, we use a subset of 5,000 examples from their training splits in each dataset. In addition, we use 1,000 examples from the NQ dev split as the development set. We

[4]While ImpRAG is general and can be adapted for iterative retrieval, we intend to focus this chapter on the single retrieval setup and will leave iterative retrieval for future work.

use the December 2021 Wikipedia from Izacard *et al.* [39] as our knowledge corpus. Additionally, we spend approximately 10% of training on plain text from Wikipedia to prevent models from overfitting to the downstream tasks.

Task	Template
Knowledge-Intensive Tasks	
NQ, Hopo, SQA, 2WQA	Q: {question} A: {answer}
AIDA	{context} Output the Wikipedia page title of the entity mentioned between [START] and [END] in the given text A: {answer}
FEV	Is this statement true? {statement} A: {answer}
T-Rex, ZsRE	{entity} [SEP] {relation} Provide the answer corresponding to the relation specified after [SEP] for the entity mentioned before [SEP] A: {answer}
Instruction-Tuning Tasks	
Dialogue Completion	{turn$_1$} {turn$_2$} {turn$_3$} ...
Reading Comprehension	{context} Q: {question} A: {answer}
Summarization	{context} Summarize this article: {summary}
Phrase Denoising	{context} Recover the original phrases marked between [START] and [END] in the given text A: {answer}
Sentence Generation	{context} [SEP] next/previous sentence Generate a sentence corresponding to the relation specified after [SEP] for the context mentioned before [SEP] A: {sentence}

Table 5.1: Prompt templates. We only use retrieval for knowledge-intensive tasks. For simplicity, we list task categories for a subset of the instruction tuning datasets. See Appendix C.4 for more detailed description.

Evaluation. We evaluate models on 8 different knowledge-intensive tasks to assess their various capabilities, specifically:

- Basic question answering: NQ, SimpleQA (SQA; [128]);
- Multihop reasoning: Hopo, 2WikiMultiHopQA (2WQA; [129]);
- Instruction following: (1) relation extraction: T-Rex [130], ZsRE [131];

  (2) fact checking: FEVER (FEV; [11]);

  (3) entity linking: AIDA [132].

For all these datasets, we report exact matches as the evaluation metric for generation tasks and recall rates for retrieval tasks. The retrieval recall is measured by the percentage of instances

	NQ	SQA	Hopo	2WQA	T-Rex	ZsRE	FEV	AIDA	avg
Llama-3.2$_{3B}$									
+RA-IT	43.2 (77.0)	38.1 (48.2)	35.9 (48.8)	33.4 (43.3)	54.2 (84.3)	58.1 (86.6)	79.2 (-)	40.1 (38.1)	47.8 (60.9)
+RA-DIT	43.4 (77.5)	38.8 (48.7)	36.4 (49.3)	34.0 (43.5)	55.0 (85.0)	59.0 (87.2)	80.5 (-)	41.0 (38.2)	48.5 (61.3)
+RA-DIT-Llama	43.9 (78.0)	39.8 **(49.9)**	37.0 (49.8)	35.1 (44.0)	55.8 (85.9)	60.0 (87.9)	80.2 (-)	41.1 (38.4)	50.4 (64.0)
+ImpRAG	**44.1 (78.4)**	**40.3 (50.0)**	**37.3 (50.2)**	**35.5 (44.5)**	**60.8 (90.2)**	**65.4 (93.2)**	**83.8 (-)**	**52.6 (58.3)**	**52.5 (66.4)**
Llama-3.1$_{8B}$									
+RA-IT	45.1 (77.0)	39.0 (48.2)	36.9 (48.8)	34.4 (43.3)	55.0 (84.3)	59.1 (86.6)	83.2 (-)	41.1 (38.1)	49.2 (60.9)
+RA-DIT	45.7 (77.7)	38.9 (48.9)	37.2 (49.1)	34.9 (44.0)	56.1 (85.4)	60.1 (87.8)	85.1 (-)	41.5 (38.8)	49.9 (61.7)
+RA-DIT-Llama	46.1 (78.7)	40.7 (50.3)	37.9 (50.2)	35.6 (44.8)	57.0 (86.1)	61.2 (88.1)	86.2 (-)	42.1 (39.2)	50.9 (62.5)
+ImpRAG	**46.4 (79.1)**	**41.3 (51.2)**	**38.4 (50.9)**	**36.0 (45.2)**	**62.5 (92.7)**	**67.1 (94.0)**	**89.2 (-)**	**54.2 (62.4)**	**54.4 (67.9)**

Table 5.2: Evaluation results for 8 knowledge-intensive tasks. We report exact match scores for generation tasks and retrieval recall (shown in parentheses) for retrieval tasks. Retrieval recall is not reported for FEV, as it is a classification task. All these methods use retrieval augmentation.

where the top-retrieved results contain the answers as substrings. We omit retrieval recall for FEV as it is a classification task where the answer strings are either “True” or “False”.

For Hopo, T-Rex, ZsRE, FEV, and AIDA, we use development sets from the KILT benchmark [7]. For SQA and NQ, we use the official test set. For 2WQA, we use their development set. For all datasets, we utilize the entire input prompts as queries for the retrievers. We describe our task templates in Table 5.1.

Baselines. We consider 3 baseline models:

- Retrieval Augmented Instruction Tuning (RA-IT): This approach involves directly incorporating retrieved passages from Contriever-MSMARCO into the context and fine-tuning the language models (LMs) on the training data;
- Retrieval Augmented Dual Instruction Tuning (RA-DIT; [38]): In this method, we first fine-tune the Contriever-MSMARCO on the training subsets of NQ and HotpotQA using Equation 5.6. Subsequently, we perform fine-tuning as in RA-IT, utilizing the fine-tuned retriever;
- RA-DIT with Llama as the Retriever (RA-DIT-Llama): Here, we replace the Contriever used in RA-DIT with the first 8 layers from the Llama models.[5] To ensure effective retrieval performance, we initially warm up the Llama retrievers with pseudo labels generated by Contriever-MSMARCO using Equation 5.3.

[5]We choose to use first 8 layers for fair comparison as ImpRAG uses the same layers for retrieval.

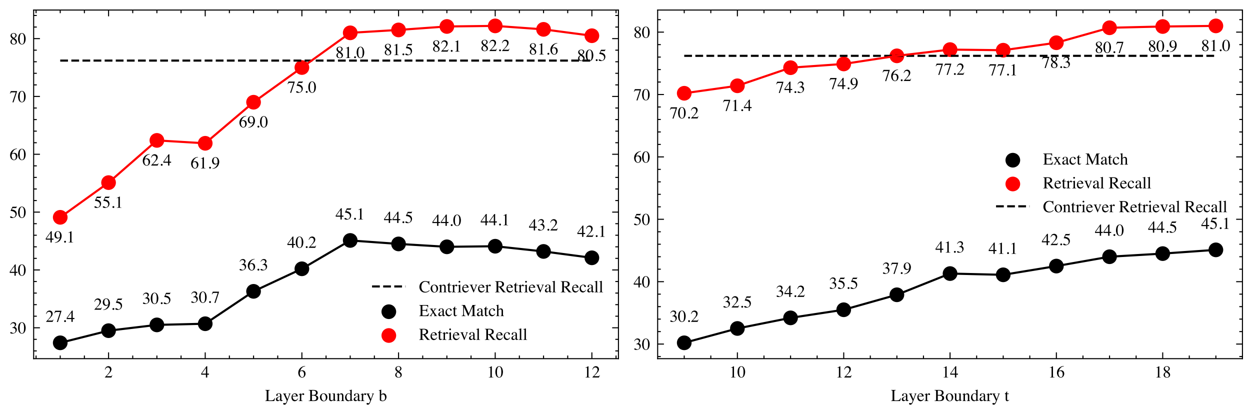


Figure 5.2: Exact match and retrieval recall on the NQ dev set using Llama-3.2 3B with different values of b (left side) and t (right side). When varying one layer boundary, we keep the other constant.

	NQ	SQA	Hopo	2WQA	T-Rex	ZsRE	FEV	AIDA	avg
self-distillation only	29.9 (61.2)	30.1 (39.9)	29.8 (41.9)	27.5 (37.4)	35.6 (64.9)	40.9 (65.9)	67.7 (-)	28.3 (22.5)	36.2 (47.7)
warmup only	**44.0 (78.3)**	39.9 **(50.0)**	37.1 (50.0)	35.1 (44.2)	56.5 (87.0)	61.2 (88.3)	81.0 (-)	45.2 (42.9)	50.0 (63.0)
warmup+self-distillation	**44.1 (78.4)**	**40.3 (50.0)**	**37.3 (50.2)**	**35.5 (44.5)**	**60.8 (90.2)**	**65.4 (93.2)**	**83.8 (-)**	**52.6 (58.3)**	**52.5 (66.4)**

Table 5.3: Exact match scores and retrieval recall (shown in parentheses) for ImpRAG using Llama-3.2 3B as the base model, trained with different retrieval objectives.

Hyperparameters. We use Llama-3.2 3B and Llama-3.1 8B as the base models for ImpRAG. For both models, the layer boundary b is set to 7.[6] For Llama-3.2 3B, the layer boundary t is 19, while for Llama-3.1 8B, it is 23. We train for 10 epochs and perform the retrieval warmup in the first 3 epochs. When retrieving passages, we take the top 10 most relevant documents.

See Appendix C.5 for more details on the baselines and computational resources.

5.4.2 Experimental Result

Table Table 5.2 presents our main evaluation results. Each model variant—RA-IT, RA-DIT, RA-DIT-Llama, and ImpRAG —exhibits different performance levels, with ImpRAG consistently achieving the highest scores across all tasks. For Llama-3.2 3B, the average exact match score increases from 47.8 with RA-IT to 52.5 with ImpRAG, while for Llama-3.1 8B, the score rises from 49.2 to 54.4. Although RA-DIT shows improvements over RA-IT, ImpRAG further en-

[6]Since we label the first layer of a LLM as layer 0, a layer boundary b of 7 means that the bottom layer group contains the first 8 layers.

	T-Rex	ZsRE	FEV	AIDA	avg
No templates	55.8 (85.9)	60.0 (87.9)	80.2 (-)	41.1 (38.4)	59.3 (70.7)
Oracle templates	**61.4 (90.7)**	**66.0 (93.6)**	**83.9 (-)**	**66.1 (72.3)**	**69.4 (85.5)**
ImpRAG	60.8 (90.2)	**65.9 (93.5)**	**83.8 (-)**	52.6 (58.3)	65.8 (80.7)

Table 5.4: Exact match scores and retrieval recall (shown in parentheses) for RA-DIT-Llama using Llama-3.2 3B as the base model, evaluated with various query templates. In the case of "no templates", the inputs to the LLMs are used directly as queries.

	NQ	SQA	Hopo	2WQA	T-Rex	ZsRE	FEV	AIDA	avg
ImpRAG	**44.1 (78.4)**	**40.3 (50.0)**	**37.3 (50.2)**	**35.5 (44.5)**	**60.8 (90.2)**	**65.4 (93.2)**	**83.8 (-)**	**52.6 (58.3)**	**52.5 (66.4)**
w/o all IT tasks	42.9 (76.4)	38.1 (48.2)	35.2 (47.7)	33.7 (42.0)	43.5 (69.3)	49.5 (70.2)	76.2 (-)	25.4 (20.5)	43.1 (53.5)
w/o PD and SG	**44.0 (78.5)**	**40.1 (50.1)**	**37.4 (50.3)**	**35.4 (44.4)**	53.3 (82.8)	57.1 (84.9)	81.2 (-)	40.5 (41.2)	48.6 (61.7)

Table 5.5: Exact match scores and retrieval recall (shown in parentheses) for ImpRAG using Llama-3.2 3B as the base model, trained with different combinations of instruction tuning datasets. IT tasks refer to instruction tuning tasks, PD stands for phrase denoising, and SG denotes sentence generation.

hances performance. Notably, ImpRAG significantly outperforms RA-DIT-Llama, indicating that the improvements are not merely due to using a more powerful base model (i.e., the first 8 layers of Llama models) for retrieval. Importantly, the enhancements are evident in both exact match scores and retrieval recalls, demonstrating that ImpRAG improves both generation quality and retrieval performance. It is worth noting that compared to the baseline approaches, the most substantial improvements with ImpRAG are seen in tasks that queries need to be formulated more differently from input prompts, such as T-Rex, ZsRE, FEVER, and AIDA. Among these tasks, AIDA shows the most significant improvements, with over a 20-point increase in retrieval recall and more than a 10-point rise in exact match scores for both Llama-3.1 3B and Llama-3.1 8B, likely due to the inadequacy of directly using input prompts as queries in AIDA. This underscores ImpRAG's effectiveness in formulating implicit queries and embedding instruction-following capabilities into retrievers. Overall, these results demonstrate that ImpRAG significantly enhances the models' ability to accurately retrieve and apply knowledge, with improvements more significant in tasks requiring diverse formats.

5.5 Analysis

5.5.1 Layer Group Boundary Ablation

In this section, we examine the effects of the layer boundaries b and t. The findings are presented in Figure 5.2. To facilitate comparison, we vary one layer boundary while keeping the other constant. We note that increasing b reduces the number of layers allocated to the middle layer group, which includes layers for reading and generation. Conversely, increasing t does not affect the retrieval layers. Overall, we find that increasing b enhances retrieval recall, with improvements leveling off once b reaches 7. This plateau is likely due to diminished generation performance, which results in less precise training signals for self-distillation. This underscores the importance of balancing parameters between retrieval and generation. On the other hand, as expected, increasing t consistently yields improvements. Although these improvements seem to plateau at 19, we refrain from further increasing t primarily due to memory constraints. We plan to leave more memory-efficient training of ImpRAG for future exploration.

5.5.2 Retrieval Objective Ablation

We conduct experiments to compare the effects of different retrieval training objectives. The results are presented in Table 5.3. During training, we consistently apply each retrieval objective throughout the entire process. For instance, in the ”warmup only” experiment, we extend the use of the warmup objective to 10 epochs instead of limiting it to the initial 3 epochs. Our findings indicate that the warmup objective provides a baseline performance across all tasks and is particularly beneficial for tasks with direct supervision. Self-distillation builds on this baseline, further enhancing model performance on unseen test tasks. Overall, the two training objectives complement each other effectively.

5.5.3 Effect of Query Templates

We also examine the impact of using different query templates for the baseline approach, RA-DIT-Llama. The results are detailed in Table 5.4. In these experiments, we omit QA tasks because their "no templates" and "oracle templates" setups are almost the same. Overall, "oracle templates" still provides the best performance. The improvements are particularly notable on AIDA. However, it is important to highlight that ImpRAG achieves highly competitive performance on 3 out of 4 tasks and already shows significant improvement on the remaining task compared to using "no templates."

5.5.4 Effect of Instruction Tuning for Retrieval

In Table 5.5, we explore the effects of training on instruction tuning datasets. The table shows that omitting all instruction tuning datasets leads to a decline in model performance on both in-domain tasks (NQ, SQA, Hopo, and 2WQA) and out-of-domain tasks. Notably, removing only phrase denoising and sentence generation has a minimal impact on in-domain tasks but causes more pronounced negative effects on out-of-domain tasks, except for FEV. This exception likely arises because FEV's task format is more similar to the in-domain tasks than other tasks. This suggests that instruction tuning tasks aid models in understanding task formats, and ImpRAG can transfer this knowledge from generation to retrieval due to its unified model architecture.

5.6 Conclusions

We present ImpRAG, a query-free retrieval-augmented generation (RAG) system that implicitly captures information needs without requiring human-specified queries. Unlike prior work that treats retrieval and generation as separate components with independently trained models, ImpRAG unifies them within a single decoder-only language model by partitioning it into specialized layer groups and jointly optimizing for both retrieval and generation. The same model parameters and forward pass are shared across retrieval and generation, effectively minimizing the mismatch between the retriever and the generator. ImpRAG demonstrates strong performance across eight knowledge-intensive tasks, outperforming traditional RAG systems and delivering substantial gains on unseen tasks with diverse formats.

Part II

Entity-Centric Language Understanding

CHAPTER 6
ENTITY LINKING AS QUESTION ANSWERING

This chapter is adapted from joint work with Wenyue Hua and Karl Stratos entitled " EntQA: Entity Linking as Question Answering" [123].

A conventional approach to entity linking is to first find mentions in a given document and then infer their underlying entities in the knowledge base. A well-known limitation of this approach is that it requires finding mentions without knowing their entities, which is unnatural and difficult. We present a new model that does not suffer from this limitation called **EntQA**, which stands for **Ent**ity linking as **Q**uestion **A**nswering. EntQA first proposes candidate entities with a fast retrieval module, and then scrutinizes the document to find mentions of each candidate with a powerful reader module. Our approach combines progress in entity linking with that in open-domain question answering and capitalizes on pretrained models for dense entity retrieval and reading comprehension. Unlike in previous works, we do not rely on a mention-candidates dictionary or large-scale weak supervision. EntQA achieves strong results on the GERBIL benchmarking platform.

6.1 Introduction

We consider the most general form of entity linking (EL) in which a system, given a document, must both extract entity mentions and link the mentions to their corresponding entries in a knowledge base (KB). EL is a foundational building block in automatic text understanding with applications to question answering (QA) [133], information retrieval [134, 135, 136, 137], and commercial recommendation systems [138, 139].

The output space in EL is intractably large. Any subset of all possible spans in the document linked to any KB entries (typically in the order of millions) can be a system output. To get around the intractability, existing methods decompose EL into mention detection (MD) and entity disambiguation (ED) and tackle them with varying degrees of independence. In all cases, however, the order of these two subproblems is MD followed by ED: first the system identifies potential entity mentions, and then the mentions are resolved to KB entries. Previous works either assume that mentions are given [46], run an off-the-shelf named-entity recognition (NER) system to extract mentions and resolve them by ED (MD$\rightarrow$ED pipeline) [47, 48, 49], or train an end-to-end model that jointly performs MD$\rightarrow$ED by beam search [8, 22].

A limitation of performing MD before ED is that it requires finding mentions without knowing the corresponding entities. By definition, a mention needs an entity (i.e., a mention of what?). Existing methods suffer from the dilemma of having to predict mentions before what they refer to, which is unnatural and difficult. For example, the MD$\rightarrow$ED pipeline heuristically extracts mentions from spans of named entities found by a third-party NER system, and the performance bottleneck is often errors in MD propagated to ED. End-to-end models alleviate the problem of error propagation, but the search is only approximate and the dilemma, albeit to a lesser degree, remains.

In this chapter, we propose flipping the order of the two subproblems and solving ED before MD. We first find candidate entities that might be mentioned in the given document, then for each candidate find its mentions if possible. Our key observation is that while finding mentions is difficult without the knowledge of relevant entities, finding relevant entities is easy without the knowledge of their specific mentions. This simple change fundamentally solves the dilemma above since identifying mentions of a particular entity is well defined.

We cast the problem as *inverted* open-domain QA. Specifically, given a document, we use a dual encoder retriever to efficiently retrieve top-K candidate entities from the KB as "questions". Then we apply a deep cross-attention reader on the document for each candidate to identify mentions of the candidate in the document as "answer spans". Unlike in standard QA, the model must predict an unknown number of questions and answers. We present a simple and effective solution based on thresholding. We call our model **EntQA**, standing for **Ent**ity linking as **Q**uestion **A**nswering.

Beyond conceptual novelty, EntQA also offers many practical advantages. First, EntQA allows us to piggyback on recent progress in dense entity retrieval and open-domain QA. For instance, we warm start EntQA with the BLINK entity retriever [67] and ELECTRA finetuned on a QA dataset [140] to obtain an easy improvement. Second, EntQA has no dependence on a hardcoded mention-candidates dictionary which is used in previous works to reduce the search space and bias the model [141, 8, 22]. The dictionary is typically constructed using a large KB-specific labeled corpus (e.g., Wikipedia hyperlinks), thus having no dependence on it makes our approach more broadly applicable to KBs without such resources. Third, training EntQA is data efficient and can be done with an academic budget, in contrast with GENRE [22] which requires industry-scale pretraining by weak supervision.

EntQA achieves strong performance on the GERBIL benchmarking platform [142]. The in-domain F_1 score on the test portion of the AIDA-CoNLL dataset is 85.8 (2.1 absolute improvement). The macro-averaged F_1 score across 8 evaluation datasets is 60.5 (2.3 absolute improvement).[1] We analyze EntQA and find that its retrieval performance is extremely strong (over 98 top-100 recall on the validation set of AIDA), verifying our hypothesis that finding relevant entities without knowing their mentions is easy. We also find that the reader makes reasonable errors such as accurately predicting missing hyperlinks or linking a mention to a correct entity that is more specific than the gold label.

6.2 Model

Let $\mathcal{E}$ denote the set of entities in a KB associated with a text title and description. Let $\mathcal{V}$ denote the vocabulary and $\mathcal{X} = \left\{x \in \mathcal{V}^T : 1 \leq T \leq T_{\max}\right\}$ the set of all documents up to length $T_{\max}$. EL is the task of mapping $x \in \mathcal{X}$ to $y \in \mathcal{P}(\mathcal{Y}(x))$ where $\mathcal{Y}(x) = \{(s, t, e) : 1 \leq s \leq t \leq |x|, e \in \mathcal{E}\}$ is the set of all possible linked spans in x and $\mathcal{P}$ is the power set. The size of the output space is $O(2^{T_{\max}^2 |\mathcal{E}|})$ where $|\mathcal{E}|$ is typically very large (e.g., around 6 million in Wikipedia) and $T_{\max}$ can also be large (e.g., > 3000 in AIDA), ruling out any naive exhaustive search as a feasible approach.

EntQA decomposes EL into two subproblems: entity retrieval and question answering. More

[1]Code available at: https://github.com/WenzhengZhang/EntQA

specifically, given a document $x \in \mathcal{X}$,

1. The **retriever** module retrieves top-K candidate entities that might be mentioned in x.

2. The **reader** module extracts mentions of each candidate entity in x (or rejects it), then returns a subset of globally reranked labeled mentions as the final prediction.

Our approach bears superficial similarities to a standard framework in open-domain QA that pipelines retrieval and span finding [31, *inter alia*], but it has the following important differences. First, instead of retrieving passages given a question, it retrieves questions (i.e., candidate entities) given a passage. Second, even when considering a single question, there can be multiple answer spans (i.e., mentions) instead of one. Both the number of gold entities present in a document and the number of mentions of each gold entity are unknown, making this setting more challenging than standard QA in which we only need to find a single answer span for a single question on a passage.

Input representation. Both the retriever and the reader work with text representations of documents and entities, thus applicable to a zero-shot setting (e.g., linking to a new KB at test time by reading entity descriptions). We use the title $\phi_{\text{title}}(e) \in \mathcal{V}^+$ and the description $\phi_{\text{desc}}(e) \in \mathcal{V}^+$ to represent an entity $e \in \mathcal{E}$. Since a document $x \in \mathcal{X}$ is generally too long to encode with a Transformer encoder which has a quadratic dependency on the input length, we break it down in $m_x \in \mathbb{N}$ overlapping passages $p_1(x) \ldots p_{m_x}(x) \in \mathcal{V}^L$ of length L with stride S (e.g., $L = 32$ and $S = 16$) and operate at the passage-level similarly as in QA [143]. When a document is long, individual passages may lose global information. For long documents, we find it beneficial to carry a document-level topical text $\psi_{\text{topic}}(x) \in \mathcal{V}^+$ across passages in that document (e.g., first sentence). We emphasize that we do *not* use any extra information outside the document. In our experiments we simply set $\psi_{\text{topic}}(x) = x_1 \in \mathcal{V}$ (i.e., the first token in the document).

Notation. We write $\textbf{enc}_S^\theta : \mathcal{V}^T \rightarrow \mathbb{R}^{d \times T}$ to denote a Transformer encoder that maps any token sequence to the same-length sequence of corresponding contextual embeddings; the symbol S is used to distinguish different encoders. We assume the usual special tokens in the input popularized by BERT [1]: `[CLS]` to represent the whole input and `[SEP]` to indicate an input boundary. We write $\oplus$ to denote the text concatenation; we insert an unused token type in the vocabulary in between two texts being concatenated. We write $M_i \in \mathbb{R}^d$ to denote the i-th column of matrix $M \in \mathbb{R}^{d \times T}$.

6.2.1 Retriever

Given a passage $p \in \mathcal{V}^+$ in document x and an entity $e \in \mathcal{E}$, the retriever computes

$$\begin{aligned} P &= \mathbf{enc}_P^\theta(\texttt{[CLS]}\, p\, \texttt{[SEP]}\, \psi_{\text{topic}}(x)) \\ E^e &= \mathbf{enc}_E^\theta(\texttt{[CLS]}\, \phi_{\text{title}}(e) \oplus \phi_{\text{desc}}(e)\, \texttt{[SEP]}) \\ \text{score}_{\text{retr}}^\theta(p, x, e) &= P_1^\top E_1^e \end{aligned}$$

At inference time, we precompute $E^e \in \mathbb{R}^d$ for each $e \in \mathcal{E}$ and use Faiss [144] for fast top-K retrieval.

Training. We train the retriever by a multi-label variant of noise contrastive estimation (NCE). Given a passage p in document x, we have a set of multiple gold entities $\mathcal{E}(p) \subset \mathcal{E}$ that are mentioned in the passage and optimize the per-example objective

$$\max_\theta \sum_{e \in \mathcal{E}(p)} \log \left(\frac{\exp\left(\text{score}_{\text{retr}}^\theta(p, x, e)\right)}{\exp\left(\text{score}_{\text{retr}}^\theta(p, x, e)\right) + \sum_{e' \in \mathbf{N}(\mathcal{E}, p)} \exp\left(\text{score}_{\text{retr}}^\theta(p, x, e')\right)} \right) \tag{6.1}$$

where $\mathbf{N}(\mathcal{E}, p) \subset \mathcal{E} \backslash \mathcal{E}(p)$ is a set of negative examples that excludes *all* gold entities $\mathcal{E}(p)$. The objective effectively constructs $|\mathcal{E}(p)|$ independent NCE instances, each of which treats a gold entity as the only correct answer while ensuring that other gold entities are not included in negative examples. We obtain 90% of $\mathbf{N}(\mathcal{E}, p)$ by sampling entities uniformly at random from $\mathcal{E} \backslash \mathcal{E}(p)$ and 10% by hard negative mining (i.e., using highest-scoring incorrect entities under the model), which is well known to be beneficial in entity retrieval [15, 67, 60].

6.2.2 Reader

Let $e_{1:K} = (e_1 \dots e_K) \in \mathcal{E}^K$ denote K candidate entities for a passage p in document x. For each $k \in \{1 \dots K\}$, the reader computes a joint encoding of (p, x, e_k) by

$$H^k = \textbf{enc}^{\theta}_{H}(\,\texttt{[CLS]}\,p \oplus \psi_{\text{topic}}(x)\,\texttt{[SEP]}\,\phi_{\text{title}}(e_k) \oplus \phi_{\text{desc}}(e_k)\,\texttt{[SEP]}\,)$$

then defines a conditional distribution over mention spans of e_k in p by

$$\begin{aligned}
p^{\theta}_{\text{start}}(s|p, x, e_k) &= \frac{\exp\left(w_{\text{start}}^{\top} H^k_s\right)}{\sum_{i=1}^{|p|+1} \exp\left(w_{\text{start}}^{\top} H^k_i\right)} && \forall s \in \{1 \dots |p|+1\} \\
p^{\theta}_{\text{end}}(t|p, x, e_k) &= \frac{\exp\left(w_{\text{end}}^{\top} H^k_t\right)}{\sum_{i=1}^{|p|+1} \exp\left(w_{\text{end}}^{\top} H^k_i\right)} && \forall t \in \{1 \dots |p|+1\} \\
p^{\theta}_{\text{span}}(s, t|p, x, e_k) &= p^{\theta}_{\text{start}}(s|p, x, e_k) \times p^{\theta}_{\text{end}}(t|p, x, e_k) && \forall s, t \in \{1 \dots |p|+1\}
\end{aligned}$$

where $w_{\text{start}}, w_{\text{end}} \in \mathbb{R}^d$ are additional parameters. The reader also multitasks reranking: it uses $w_{\text{rerank}} \in \mathbb{R}^d$ to define a conditional distribution over candidate entities by

$$p^{\theta}_{\text{rerank}}(e_k|p, x, e_{1:K}) = \frac{\exp\left(w_{\text{rerank}}^{\top} H^k_1\right)}{\sum_{k'=1}^{K} \exp\left(w_{\text{rerank}}^{\top} H^{k'}_1\right)} \qquad \forall k \in \{1 \dots K\}$$

Training. We obtain candidates $e_{1:K}$ from a fully trained retrieval module to make training consistent with test time. During training, we always include all gold entities as candidates (i.e., $\mathcal{E}(p) \subset e_{1:K}$). Let $\mathcal{M}(p, e)$ denote the set of gold mention spans of $e \in \mathcal{E}$ in p; if e is not present in p, we define $\mathcal{M}(p, e) = \{(1, 1)\}$. We optimize the per-example objective

$$\max_{\theta} \sum_{k=1}^{K} \mathbb{1}(e_k \in \mathcal{E}(p)) \log p^{\theta}_{\text{rerank}}(e_k|p, x, e_{1:K}) + \sum_{(s,t) \in \mathcal{M}(p, e_k)} \log p^{\theta}_{\text{span}}(s, t|p, x, e_k) \tag{6.2}$$

where $\mathbb{1}(A)$ is the indicator function equal to one if A is true and zero otherwise. Note that the reader is trained to predict the `[CLS]` span for incorrect entities.

6.2.3 Inference

At test time, we process a new document $x \in \mathcal{X}$ in passages $p \in \mathcal{V}^L$ independently as follows:

1. Retrieve top-K highest scoring entities $e_{1:K}$ under $\mathrm{score}^{\theta}_{\mathrm{retr}}(p, x, e)$.
2. For each candidate k, extract top-P most likely mention spans $(s^k_1, t^k_1) \dots (s^k_P, t^k_P)$ under $p^{\theta}_{\mathrm{span}}(s, t|p, x, e_k)$ while discarding any span less probable than $(1, 1)$.
3. Return a subset of the surviving labeled mentions (s, t, e_k) with $p^{\theta}_{\mathrm{rerank}}(e_k|p, x, e_{1:K}) \times p^{\theta}_{\mathrm{span}}(s, t|p, x, e_k) > \gamma$ as the final prediction.

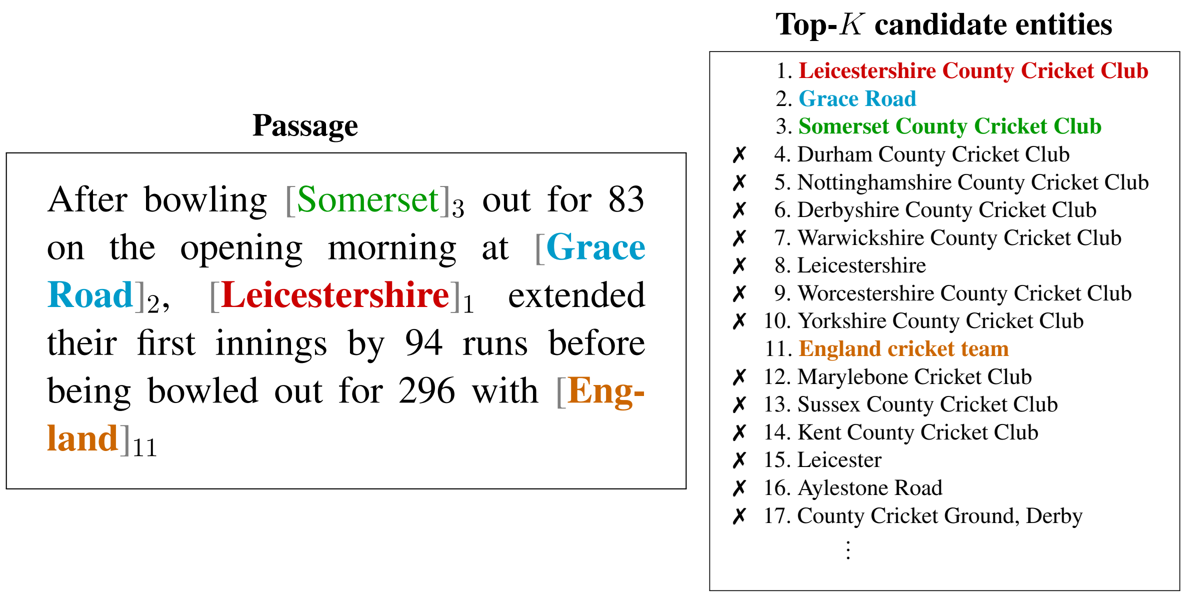
We do not apply any further processing to combine passage-level predictions other than merging duplicate labeled spans (s, t, e) in the overlapping sections. This inference scheme is simple yet effective. For each candidate entity, the reader scrutinizes the passage with deep cross-attention to see if there are any mentions of the entity and has a chance to reject it by predicting $(1, 1)$. The reader delays its final decision until it has processed all candidates to globally reconsider labeled mentions with ranking probabilities. Figure 6.1 shows a successful prediction on a passage from the validation portion of AIDA.

6.3 Experiments

We evaluate EntQA on the GERBIL benchmarking platform [142], which offers reliable comparison with state-of-the-art EL methods on numerous public datasets.

6.3.1 Setting

Datasets. We follow the established practice and report the InKB Micro F_1 score on the in-domain and out-of-domain datasets used in [22]. Specifically, we use the AIDA-CoNLL dataset [47] as the in-domain dataset: we train EntQA on the training portion of AIDA, use the validation portion (AIDA-A) for development, and reserve the test portion (AIDA-B) for in-domain test performance. We use seven out-of-domain test sets: MSNBC, Derczynski (Der) [145], KORE 50



Passage

After bowling [Somerset]$_3$ out for 83 on the opening morning at [**Grace Road**]$_2$, [**Leicestershire**]$_1$ extended their first innings by 94 runs before being bowled out for 296 with [**England**]$_{11}$

Top-K candidate entities

	1. **Leicestershire County Cricket Club**
	2. **Grace Road**
	3. **Somerset County Cricket Club**
✗	4. Durham County Cricket Club
✗	5. Nottinghamshire County Cricket Club
✗	6. Derbyshire County Cricket Club
✗	7. Warwickshire County Cricket Club
✗	8. Leicestershire
✗	9. Worcestershire County Cricket Club
✗	10. Yorkshire County Cricket Club
	11. **England cricket team**
✗	12. Marylebone Cricket Club
✗	13. Sussex County Cricket Club
✗	14. Kent County Cricket Club
✗	15. Leicester
✗	16. Aylestone Road
✗	17. County Cricket Ground, Derby
	⋮

Figure 6.1: Example prediction by EntQA taken from AIDA-A. Given a passage, the retriever module ranks K candidate entities, then the reader module finds mentions of each entity or rejects it (marked by ✗). Both modules use entity descriptions (not shown). In this example, it predicts the span "England" for the 11th candidate `England cricket team` but rejects the 35th candidate `England` (the country).

(K50) [146], N3-Reuters-128 (R128), N3-RSS-500 (R500) [147], and OKE challenge 2015 and 2016 (OKE15 and OKE16) [148]. We refer to Table 6 in [8] for the datasets' statistics. For the KB, we use the 2019 Wikipedia dump provided in the KILT benchmark [149], which contains 5.9 million entities.

Model details. We initialize the passage encoder $\mathbf{enc}_P^\theta$ and the entity encoder $\mathbf{enc}_E^\theta$ in the retriever module with independent BLINK retrievers pretrained on Wikipedia hyperlinks [67] and optimize the NCE objective (6.1) with hard negative mining. We initialize the joint encoder $\mathbf{enc}_H^\theta$ in the reader module with ELECTRA-large [140] finetuned on SQuAD 2.0 [150] and optimize the reader objective (6.2). We break up each document $x \in \mathcal{X}$ into overlapping passages of length $L = 32$ with stride $S = 16$ under WordPiece tokenization. For each passage in x, we concatenate the input with the first token of the document $\psi_{\text{topic}}(x) = x_1$, which corresponds to the topic in AIDA but not in other datasets. We use 64 candidate entities in training for both the retriever and the reader; we use 100 candidates at test time. We predict up to $P = 3$ mention spans for each candidate entity. We use $\gamma = 0.05$ as the threshold in all experiments, chosen after trying values 0.01, 0.1, and 0.05 on the validation set. Additional experiments on automatically tuning γ are discussed in Appendix D.1. For optimization, we use Adam [96] with learning rate 2e-6 for the retriever and 1e-5 for the reader; we use a linear learning rate decay schedule with warmup proportion 0.06 over 4 epochs for both modules. The batch size is 4 for the retriever and 2 for the reader. The retriever is trained on 4 GPUs (A100) for 9 hours; the reader is trained on 2 GPUs for 6 hours.

Baselines. We compare with state-of-the-art EL systems that represent a diverse array of approaches. [47] and [49] use the MD$\rightarrow$ED pipeline; despite the limitation of pipelining MD with ED, the latter achieve excellent performance by solving MD with a strong NER system [151]. [8] use an end-to-end model that sequentially performs MD and ED; to make the problem tractable, they drastically prune the search space with a mention-candidates dictionary and the model score. [22] propose GENRE, a sequence-to-sequence model for EL. The model conditions on the given document and autoregressively generates a labeled version of the document by at each position either copying a token, starting or ending a mention span, or, if the previous generation was the end of a mention m, generating the entity title associated with m token by token. At inference time, GENRE critically relies on a prefix tree (aka. trie) derived from Wikipedia to constrain the beam search so that it produces a valid entity title in the KB. Since each beam element must first predict a mention before predicting an entity, unless the beam size is unbounded so that every labeled span is considered, GENRE will suffer from MD errors propagating to ED.

6.3.2 Results

Table 6.1 shows the main results. EntQA achieves the best in-domain test F_1 score for AIDA (+2.1) and is also performant on out-of-domain datasets (+3.8 on KORE 50 and +7.4 on N3-Reuters-128, close second-best on Derczynski and N3-RSS-500). The performance is lower on OKE15 and OKE16 for the same reason pointed out by [22]: these datasets are annotated with coreference (i.e., they contain pronouns and common nouns linked to entities) which our model is

Table 6.1: InKB Micro F_1 on the in-domain and out-of-domain test sets on the GERBIL benchmarking platform. For each dataset, **bold** indicates the best model and <u>underline</u> indicates the second best.

	In-domain	Out-of-domain							
Method	**AIDA**	**MSNBC**	**Der**	**K50**	**R128**	**R500**	**OKE15**	**OKE16**	**Avg**
Hoffart *et al.* [47]	72.8	65.1	32.6	55.4	46.4	**42.4**	**63.1**	0.0	47.2
Steinmetz and Sack [152]	42.3	30.9	26.5	46.8	18.1	20.5	46.2	46.4	34.7
Moro *et al.* [153]	48.5	39.7	29.8	55.9	23.0	29.1	41.9	37.7	38.2
Kolitsas *et al.* [8]	82.4	<u>72.4</u>	34.1	35.2	<u>50.3</u>	38.2	<u>61.9</u>	<u>52.7</u>	53.4
Broscheit [154]	79.3	-	-	-	-	-	-	-	
Martins *et al.* [155]	81.9	-	-	-	-	-	-	-	
Hulst *et al.* [49]	80.5	<u>72.4</u>	41.1	50.7	49.9	35.0	**63.1**	**58.3**	56.4
De Cao *et al.* [22]	<u>83.7</u>	**73.7**	**54.1**	<u>60.7</u>	46.7	40.3	56.1	50.0	<u>58.2</u>
EntQA	**85.8**	72.1	<u>52.9</u>	**64.5**	**54.1**	<u>41.9</u>	61.1	51.3	**60.5**

not trained for, while many other systems have a component in their pipelines to handle these cases. We hypothesize that the performance on MSNBC is lagging because it has long documents (544 words per document on average) which are processed in relatively short passages under EntQA due to our computational constraints. Overall, EntQA achieves the best macro-averaged F_1 score across the 8 evaluation datasets (+2.3).

The inference runtime of EntQA is clearly linear in the number of candidate entities K. To get a sense of speed, we compared the runtime of EntQA with that of GENRE on the AIDA validation set using 1 GPU on the same machine. GENRE took 1 hour and 10 minutes, excluding 31 minutes to first build a prefix tree. EntQA took 20 minutes with $K = 100$, 10 minutes with $K = 50$, and 4 minutes with $K = 20$, excluding 1 hour to first index entity embeddings, yielding F_1 scores 87.3, 87.4, and 87.0. Interestingly, we can obtain a significant speedup at a minor cost in performance by decreasing K. We believe this can be a useful feature of the model in controlling the speed-performance tradeoff.

We note that there is an issue of using different editions of Wikipedia between the systems. For instance, [47] use the 2010 dump, [49] and we use the 2019 dump, whereas [8] and [22] use the

2014 dump (even though the latter use the 2019 dump for pretraining). Thus there is a concern that differences in performance are due to different snapshots of Wikipedia. While we consider it out of scope in our work to fully address this concern, we find that using different editions of Wikipedia does not fundamentally change the performance of EntQA, which is consistent with GERBIL's intent of being KB-agnostic. For instance, we obtained the same validation F_1 on AIDA with our model trained on either the 2014 or 2019 dump. We use the KILT edition of Wikipedia mainly for convenience.

6.3.2.1 Other Practical Highlights

No dictionary. EntQA has no dependence on a mention-candidates dictionary. All previous works rely on a dictionary $\mathcal{D} : \mathcal{V}^+ \rightarrow \mathcal{P}(\mathcal{E})$ that maps a mention string m to a small set of candidate entities $e \in \mathcal{E}$ associated with empirical conditional probabilities $\hat{p}_{e|m} > 0$ [47, *inter alia*]. For instance, it is an essential component of the search procedure in the end-to-end model of [8]. While not mentioned in the paper or on the GitHub repository, GENRE [22] also uses the dictionary from [8] in their prefix tree to constrain the beam search (personal communication with one of the authors of the paper). Constructing such a dictionary typically assumes the existence of a large KB-specific labeled corpus (e.g., internal links in Wikipedia). EntQA is thus more broadly applicable to KBs without such resources (e.g., for small domain-specific KBs).

No model-specific pretraining. EntQA does not require model-specific pretraining; it only uses standard pretrained Transformers for initialization and is directly finetuned on AIDA. This is in contrast with GENRE which requires industry-scale pretraining by weak supervision. Specifically, GENRE is trained by finetuning BART [35] on autoregressive EL training examples constructed from all Wikipedia abstract sections on 64 GPUs for 30 hours, followed by finetuning on AIDA. Thus training GENRE from scratch is beyond the means of most academic researchers, making it difficult to make substantial changes to the model. EntQA can be trained with academic resources and outperforms GENRE.

6.3.3 Ablation Studies

The final form of EntQA in Section 6.3.2 is the result of empirically exploring various modeling and optimization choices during development. We present an ablation study to illustrate the impact of these choices.

Retriever Table 6.2 shows an ablation study for the retriever module. We report top-100 recall (R@100) on the validation set of AIDA. The baseline retriever is initialized with BLINK [67], uses the passage representation `[CLS]`p`[SEP]`x_1, and is trained by optimizing the multi-label variant of NCE (6.1) that considers one gold entity at a time by excluding others in the normalization term. We see that the baseline retriever has an extremely high recall (98.2), confirming our hypothesis that it is possible to accurately infer relevant entities in a passage without knowing where they are mentioned. We also see that it is very important to use the proposed multi-label variant of NCE instead of naive NCE that normalizes over all gold entities, which results in a massive decrease in recall (82.7). We consider optimizing the marginal log-likelihood (i.e., the log of the sum of the probabilities of gold entities, rather than the sum of the log), but it yields much worse performance (83.8). It is helpful to initialize with BLINK rather than BERT-large, use hard negatives in NCE, and append x_1 to input passages. Table 6.2 additionally shows the BM25 recall, which is quite poor (36.6). Upon inspection, we find that BM25 fails to retrieve diverse entities. For instance, a passage on cricket may have diverse gold entities such as an organization (`Leicestershire County Cricket Club`), location (`London`), and person (`Phil Simmons`), but the top entities under BM25 are dominated by person entities (`Alan Shipman`, `Dominique Lewis`, etc.). This shows the necessity of explicitly training a retriever to prioritize diversity in our problem.

Reader Table 6.3 shows an ablation study for the reader module. We report F_1 on the validation set of AIDA. The baseline reader is initialized with ELECTRA-large [140] finetuned on SQuAD 2.0, uses the joint passage-entity input representation `[CLS]`$p \oplus x_1$`[SEP]`$\phi_{\text{title}}(e) \oplus \phi_{\text{desc}}(e)$`[SEP]`, and is trained by optimizing (6.2). Candidate entities are obtained from the base-

Table 6.2: Ablation study for the retriever module. Each line makes a single change from the baseline retriever used in Table 6.1. We also compare with BM25.

Retriever	Val R@100
Baseline	98.2
– Omit excluding other gold entities in the normalization term of NCE	82.7
– Train by optimizing the marginal log-likelihood	83.8
– Initialize with BERT-large	94.4
– Omit hard negatives in NCE (i.e., negative examples are all random)	94.4
– Omit the document-level information x_1 in the passage representation	96.6
BM25	36.6

Table 6.3: Ablation study for the reader module. Each line makes a single change from the baseline reader used in Table 6.1. Candidate entities are obtained from the baseline retriever in Table 6.2 (except the oracle experiment).

Reader	Val F_1
Baseline	87.3
– Initialize with BERT-large	85.6
– Train by optimizing the marginal log-likelihood	86.9
– Initialize with ELECTRA-large (not finetuned on SQuAD 2.0)	88.4
– Omit the reranking probabilities $p^{\theta}_{\text{rerank}}$ (i.e., only use span probabilities)	87.9
– Omit the document-level information x_1 in the input passage representation	87.5
Oracle experiment: use gold entities as the only candidate entities	94.9

line retriever in Table 6.2. We see that BERT is less performant than ELECTRA for reader initialization, consistent with findings in the QA literature [156]. Training by optimizing the marginal log-likelihood is comparable to (6.2). Interestingly, we find that we can fit the reader just as well without using a SQuAD-finetuned ELECTRA, ranking probabilities, or x_1 in passages. However, in our preliminary investigation we found that these variants generalized slightly worse outside the training domain, thus we kept our original choice. We discuss other choices of document-level information in Appendix D.2. Lastly, we conduct an oracle experiment in which we provide only gold entities as candidates to the reader. In this scenario, the reader is very accurate (94.9 F_1), suggesting that the main performance bottleneck is correctly distinguishing gold vs non-gold entities

Table 6.4: Categorizing errors on the validation set passages. The number of passages in each category is given in parentheses. **G** refers to the gold annotation; **P** refers to the predicted annotation.

Error	Examples (text snippets)	
Over	**G**: england fast bowler [martin mccague]$_{\text{Martin McCague}}$	(Fill in missing mentions)
(443)	**P**: [england]$_{\text{England cricket team}}$ fast bowler [martin mccague]$_{\text{Martin McCague}}$	
	G: duran, 45, takes on little - known [mexican]$_{\text{Mexico}}$	
	P: [duran]$_{\text{Roberto Durán}}$, 45, takes on little - known [mexican]$_{\text{Mexico}}$	
Under	**G**: second innings before [simmons]$_{\text{Phil Simmons}}$ stepped in	(Bad threshold)
(474)	**P**: second innings before simmons stepped in	
	G: [ato boldon]$_{\text{Ato Boldon}}$ - lpr - [trinidad]$_{\text{Trinidad}}$ - rpr - 20.	
	P: [ato boldon]$_{\text{Ato Boldon}}$ - lpr - trinidad - rpr - 20.	
Neither	**G**: match against yorkshire at [headingley]$_{\text{Headingly}}$	(Ambiguous entity)
(378)	**P**: match against yorkshire at [headingley]$_{\text{Headingly Stadium}}$	
	G: at the [oval]$_{\text{The Oval}}$, surrey captain chris lewis	(Ambiguous span)
	P: at [the oval]$_{\text{The Oval}}$, surrey captain chris lewis	
	G: scores in [english]$_{\text{England}}$ county championship matches	(Others)
	P: scores in [english county championship]$_{\text{County Championship}}$ matches	

from the candidates. We investigate this issue more in depth in the next section.

6.3.4 Error Analysis

To better understand the source of errors made by EntQA, we examine passages in the validation set for which the model's prediction is not completely correct. We partition them into three types: (1) over-predicting (i.e., the gold mentions are a strict subset of the predicted mentions), (2) under-predicting (i.e., the predicted mentions are a strict subset of the gold mentions), and (3) neither over- nor under-predicting. Table 6.4 shows examples of each error type. We find that over-predicting often happens because the model correctly "fills in" entity mentions missing in the gold annotation. Under-predicting happens most likely because the threshold value is too large to catch certain mentions. Finally, many errors that are neither over- nor under-predicting are largely due to annotation noise. For instance, the predicted entity `Headingly Stadium` is a correct and more specific entity for the span "headingley" than the gold entity `Headingly` (a suburb); the predicted span "the oval" is more suitable, or at least as correct as, the gold span "oval" for the entity `The Oval`.

We also consider distinguishing MD errors from ED errors on the validation set. EntQA obtains 87.5 overall F_1. When we only measure the correctness of mention spans (equivalent to treating all entity predictions as correct), we obtain 92.3 F_1. When we only measure the correctness of rejecting or accepting candidate entities, we obtain 64.5 F_1 at the passage level and 89.3 F_1 at the document level (i.e., consider the set of candidates from all passages). The reader's relatively low passage-level F_1 in rejecting or accepting candidates is consistent with the the oracle experiment in Table 6.3. That is, the main performance bottleneck of EntQA is discriminating gold vs non-gold entities from the candidates, though this should be taken with a grain of salt given the noise in annotation illustrated in Table 6.4.

6.4 Related Work

Our work follows the recent trend of formulating language tasks as QA problems, but to our knowledge we are the first to propose reduction to inverted open-domain QA. Most previous works supply questions as input to the system, along with passages in which answer spans are found. They differ only in question formulation, for instance a predicate in semantic role labeling [157], a relation type along with its first argument in KB completion [158, 159], an entity category in (nested) NER [160], an auxiliary verb or a *wh*-expression in ellipsis resolution [161], and other task-specific questions [162]. In contrast, we solve question formulation as part of the problem by exploiting recent advances in dense text retrieval.

A notable exception is CorefQA [163], from which we take direct inspiration. In this approach, the authors formulate coreference resolution as QA in which questions are coreferring spans and answers are the spans' antecedents (i.e., earlier spans that belong to the same coreference cluster). Since coreferring spans are unknown, the authors rely on the end-to-end coreference resolution model of [9] that produces candidate spans by beam search. In contrast, EntQA handles varying numbers of questions in a simpler framework of text retrieval.

As in this chapter, some previous works propose methods to handle varying numbers of answer spans for a given question. But their methods are based on one-vs-all classification (i.e., each label is associated with a token-level binary classifier) or reduction to tagging (i.e., spans are expressed as a BIO-label sequence) [163, 159, 160]. We found these methods to be ineffective in preliminary experiments, and instead develop a more effective inference scheme in which the model delays its final prediction to the end for global reranking (Section 6.2.3).

We discuss pros and cons of EntQA vs other models in practice. While EntQA outperforms GENRE without large-scale weakly supervised pretraining, it involves dense retrieval which incurs a large memory footprint to store and index dense embeddings as pointed out by [22]. But it can be done on a single machine with ample RAM (ours has 252G) which is cheap. Bypassing dense retrieval is a unique strength of the autoregressive approach of GENRE and orthogonal to ours; we leave combining their strengths as future work. Our model requires a threshold γ for inference, but we find that it is easy to pick a good threshold; we also argue that it can be a useful feature in a real-world setting in which the practitioner often needs a customized trade-off between precision and recall. The threshold-based inference implies another unique feature of EntQA not explored in this chapter: it can naturally handle nested entity mentions. We leave nested linking as future work.

6.5 Conclusions

Existing methods for entity linking suffer from the dilemma of having to predict mentions without knowing the corresponding entities. We have presented EntQA, a new model that solves this dilemma by predicting entities first and then finding their mentions. Our approach is based on a novel reduction to inverse open-domain QA in which we retrieve an unknown number of questions (candidate entities) and predict potentially multiple answer spans (mentions) for each question. Our solution is a simple pipeline that takes full advantage of progress in text retrieval and reading comprehension. EntQA achieves new state-of-the-art results on the GERBIL benchmarking platform without relying on a KB-specific mention-candidates dictionary or expensive model-specific pretraining.

CHAPTER 7
SEQUENCE-TO-SEQUENCE COREFERENCE RESOLUTION

This chapter is adapted from joint work with Sam Wiseman and Karl Stratos entitled " Seq2seq is All You Need for Coreference Resolution" [164].

Existing works on coreference resolution suggest that task-specific models are necessary to achieve state-of-the-art performance. In this chapter, we present compelling evidence that such models are not necessary. We finetune a pretrained seq2seq transformer to map an input document to a tagged sequence encoding the coreference annotation. Despite the extreme simplicity, our model outperforms or closely matches the best coreference systems in the literature on an array of datasets. We also propose an especially simple seq2seq approach that generates only tagged spans rather than the spans interleaved with the original text. Our analysis shows that the model size, the amount of supervision, and the choice of sequence representations are key factors in performance.

7.1 Introduction

The seminal work by Lee *et al.* [165] popularized end-to-end models for coreference resolution based on searching over all possible spans and their clustering. However, even with substantial refinement and simplification in followup works [50, 51, 163, 52], the models are highly task-specific, involving many specialized hyperparameters such as how many candidates to retain in the mention proposal phase.

There is a recent line of works that take an alternative approach, leveraging advances in pretrained sequence-to-sequence (seq2seq) models. Liu *et al.* [53] propose ASP, an autoregressive pointer-based model with a multitasking head for bracket pairing and span labeling. Bohnet *et al.* [54] propose a transition-based system with carefully designed states and actions, simulated by a seq2seq model that processes one state at a time. However, these seq2seq-based models are *still* task-specific, requiring a modification of the seq2seq architecture or a derivation of a transition-based system with state manipulation.

A natural question is: are such task-specific models necessary for coreference resolution, or can we approach it as a standard seq2seq problem? There have been previous efforts to reduce coreference resolution as a seq2seq problem [55, 56], but they underperform task-specific models, suggesting that task-specific models are perhaps necessary.

In this chapter, we present the first full seq2seq reduction of coreference resolution that matches or outperforms the best coreference systems in the literature, demonstrating that task-specific models are not necessary to obtain state-of-the-art performance and questioning the need to develop task-specific solutions. Our approach is extremely simple. We treat the raw document as a source sequence and the coreference annotation as a target sequence, then finetune a pretrained encoder-decoder model like T5 [166] or T0 [167] without any modification to the architecture.

There is a great deal of flexibility in the choice of target sequence representation. Our main model represents the coreference annotation as a sequence of actions that either copy a token from the source sequence, start/end a mention span, or tag a predicted mention span with an integer. At test time, the model always produces a valid coreference clustering by constrained beam search. We consider an even simpler version in which the model generates only the tagged spans, and find that it also yields surprisingly high performance. This simpler version is advantageous because it results in faster inference.

Our seq2seq model obtains strong results on an array of coreference datasets. On English OntoNotes [168], with a 3B-parameter T0 the model obtains 82.9 test average F1, outperforming the corresponding ASP [53] initialized from the same base model (82.3). With a 11B-parameter T0, our model achieves 83.2 F1, outperforming CorefQA [163] (83.1) and getting close to the best known result using a 13B-parameter model (83.3). On PreCo [169], the model achieves 88.5, outperforming the task-specific model of Toshniwal *et al.* [170] (87.8). On LitBank [171], which has substantially smaller training data, the model obtains 78.3 cross-validation F1 lagging behind Toshniwal *et al.* [170] who report 79.2. But when trained on the union of LitBank, OntoNotes, and PreCo, it obtains 81.2 split-0 F1, significantly outperforming their 78.2. Our analysis shows that the model size, the amount of supervision, and the choice of sequence representations are key factors in performance. We make our code publicly available at: https://github.com/WenzhengZhang/Seq2seqCoref.

7.2 Related Work

In this section, we give a more focused treatment of previous works on seq2seq-style approaches to coreference resolution to make our contributions more clear. Urbizu *et al.* [55] propose framing coreference resolution as a seq2seq task, but their work is a proof of concept. They naively predict boundaries, cluster numbers, and blanks (e.g., "(0 – – 0) – (1 – (2) — 1)") with suboptimal modeling choices (e.g., their decoder only receives these symbols and no document content). They only report results on the ARRAU corpus and significantly fall behind existing works (e.g., 66.5 vs 78.8 under B^3). In contrast, we solve a much more challenging problem of developing state-of-the-art seq2seq coreference systems.

There are recent works approaching structured prediction with a general seq2seq-style solution. One example is TANL [56], which frames entity/relation extraction, semantic role labeling, and coreference resolution as seq2seq tasks. Again, TANL fails to demonstrate competitive performance on standard coreference resolution datasets, obtaining only 72.8 average F1 on OntoNotes (compared to the current state-of-the-art performance level which is > 80). Their target sequence representation corresponds to our full linearization with token action except that they tag the cluster information by preceding mention strings (e.g., "[his — Barack Obama]"), which yields long target sequences and introduces clustering ambiguity. While we improve the performance of TANL to 79.6 in our own implementation, achieving the best performance (83.2) requires our sequence definitions (Section 7.3).

ASP [53] is an autoregressive pointer-based model for structured prediction. It is a modified transformer that at step t conditions on the input document x and a sequence representation of the annotation so far $z_{\leq t}$ and predicts a tuple of model-specific actions $(\alpha_t, \beta_t, \gamma_t)$ used for structure building. For instance, $\beta_t \in \{0 \dots t-1\}$ is a bracket pairing action parameterized with a feedforward layer that consumes the previous hidden states (h_t, h_{β_t}) (i.e., a pointer network). ASP obtains state-of-the-art results on OntoNotes (up to 82.5 average F1). We outperform ASP with a standard transformer.

Bohnet *et al.* [54] develop a transition-based coreference system that can be implemented by a seq2seq model. The system autoregressively maps a state to a prediction, where a state is previous coreference-annotated sentences along with the next sentence and a prediction is system actions (e.g., link and append). While the system can be reduced to seq2seq predictions, it processes one sentence at a time by applying the predicted actions to the current state. We show that such an explicit state-by-state transition is not necessary, and that a standard transformer can directly reason with all coreference clusters simultaneously.

7.3 Seq2Seq Methods

Let $\mathcal{V}$ denote the vocabulary. For any sequence $a \in \mathcal{V}^T$ and position $t \in \{1 \dots T\}$, we use the prefix notation $a_{<t} = (a_1 \dots a_{t-1}) \in \mathcal{V}^{t-1}$ and $a_{\leq t} = (a_1 \dots a_t) \in \mathcal{V}^t$. We assume a generalized seq2seq setting in which $x \in \mathcal{V}^{T'}$ is the source sequence, $y \in \mathcal{V}^T$ is the target sequence (i.e., a sequence of labels), and $z \in \mathcal{V}^T$ is an additional sequence fed to the seq2seq decoder where $z_t = F(x, z_{<t}, y_{<t})$ is some fixed deterministic mapping. The "generalized" model uses parameters θ to define the conditional distribution

$$p_\theta(y|x) = \prod_{t=1}^{T} p_\theta(y_t|x, z_{\leq t}). \tag{7.1}$$

We emphasize that the generalization above does not modify the standard seq2seq framework. During training, we feed (x, z) to the model as the usual source-target sequence pair and minimize the cross-entropy loss using y as per-token labels. At test time, we apply the mapping F at each step by postprocessing predictions.

All our methods use (7.1) and only change the variable definitions. The encoder input x is always the document to be processed. The decoder input z is a sequence representation or "linearization" of the coreference annotation of x. The decoder output y is any action sequence from which z can be extracted deterministically at each step.

7.3.1 Linearization of the Coreference Annotation

For a document $x \in \mathcal{V}^{T'}$, the coreference annotation S is a set of spans clustered into C groups formalized as

$$S \subset \{(i, j, l) : \ 1 \leq i \leq j \leq T', \ 1 \leq l \leq C\}$$

Note that the spans can be nested. We assume that the spans are ordered in non-decreasing lengths. We will use the following as a running example:

$$\begin{aligned} x &= (a, b, c, d, e) \\ S &= \{(2, 2, 1), (5, 5, 2), (2, 3, 2)\} \end{aligned} \tag{7.2}$$

(i.e., the clustered spans are $\{\{b\}, \{(b, c), e\}\}$). The goal is to express S as a sequence $z \in \mathcal{V}^T$ for some length T. A minimal formulation is to literally predict the integer triples, for instance $z = \texttt{str}(S)$ where $\texttt{str}$ is the string conversion in Python. However, while short, such a non-linguistic representation was found to perform poorly likely because it is not compatible with language model pretraining.

A better approach is to predict the mentions. We represent the tagged span (i, j, l) in document x as

$$\texttt{<m>} \ x_i \dots x_j \mid l \ \texttt{</m>}$$

where $\texttt{<m>}, \texttt{</m>} \in \mathcal{V}$ are special symbols indicating the start and end of a mention representation, and $\mid \in \mathcal{V}$ indicates the end of a mention string. Nested spans are handled naturally by linearizing the spans in order (i.e., from the shortest to the longest). For instance, the subsequence $(b, c) \in \mathcal{V}^2$

in the running example (7.2) will become the length-10 sequence

$$\texttt{<m>}\ \texttt{<m>}\ b \mid 1\ \texttt{</m>}\ c \mid 2\ \texttt{</m>}$$

In contrast to a transition-based approach [54], we decode all coreference clusters in the document in a single pass. Thus there is an issue of alignment: if the model predicts multiple mentions with the same string form, how do we know the corresponding spans in the source sequence? A simple solution popular in general seq2seq approaches to tagging is to exhaustively predict all the tokens in x [172, 22]. The example (7.2) is then linearized as

$$a\ \texttt{<m>}\ \texttt{<m>}\ b \mid 1\ \texttt{</m>}\ c \mid 2\ \texttt{</m>}\ d\ \texttt{<m>}\ e \mid 2\ \texttt{</m>}$$

We call this representation **full linearization**. Full linearization completely eliminates the problem of alignment ambiguity at the cost of longer target sequences. We also consider an alternative shorter representation that we call **partial linearization** in which only the tagged mentions are encoded. In this case, (7.2) is linearized as

$$\texttt{<m>}\ \texttt{<m>}\ b \mid 1\ \texttt{</m>}\ c \mid 2\ \texttt{</m>}\ \texttt{<m>}\ e \mid 2\ \texttt{</m>}$$

Partial linearization has the potential to drastically shorten the target length if the mentions are sparse, but it requires an explicit alignment step to transfer the predictions to the input document. We defer the discussion of the alignment problem to Section 7.3.4.

7.3.2 Action Sequences

Given a choice of linearization $z \in \mathcal{V}^T$ (i.e., input to the decoder), we can choose any action sequence $y \in \mathcal{V}^T$ (i.e., the actual predictions by the decoder) such that at each step t, we can extract z_t from the document x and the past information $z_{<t}$ and $y_{<t}$. We assume that $z_1 = \texttt{<s>}$ and $z_{T+1} = \texttt{</s>}$ are the standard start and end symbols for the target sequence. A straightforward action sequence is given by $y_t = z_{t+1}$ for $t = 1 \ldots T$ which we call **token action**. Token action corresponds to a common language modeling setting where the per-step prediction is simply the next token. For instance, the annotation $x = (a, b)$ and $S = \{(2, 2, 1)\}$ under full linearization and token action is assigned

$$\begin{aligned} y &= (a, \texttt{<m>}, b, |, 1, \texttt{</m>}, \texttt{</s>}) \\ z &= (\texttt{<s>}, a, \texttt{<m>}, b, |, 1, \texttt{</m>}, \texttt{</s>}) \end{aligned} \tag{7.3}$$

Under full linearization, we can use a smaller action space $\mathcal{A} = \{\texttt{<c>}, \texttt{<m>}, \texttt{</m>}, |, \texttt{</s>}\} \cup \mathcal{U}$ where <c> is the special "copy" symbol which means copying a single token from the document and advancing the index by one, and $\mathcal{U} \subset \mathcal{V}$ is the subset of the vocabulary used for encoding integers. We call this choice **copy action**, formally defined as

$$y_t = \begin{cases} \texttt{<c>} & \text{if } z_{t+1} \notin \{\texttt{<m>}, \texttt{</m>}, |, \texttt{</s>}\} \cup \mathcal{U} \\ z_{t+1} & \text{otherwise} \end{cases}$$

The example (7.3) under copy action becomes

$$\begin{aligned} y &= (\texttt{<c>}, \texttt{<m>}, \texttt{<c>}, |, 1, \texttt{</m>}, \texttt{</s>}) \\ z &= (\texttt{<s>}, a, \texttt{<m>}, b, |, 1, \texttt{</m>}, \texttt{</s>}) \end{aligned}$$

Copy action in conjunction with constrained beam search is a natural way to prevent deviations between the source and target sequences [172]. Partial linearization is not compatible with copy action since the model skips the gap between mentions.

7.3.3 Integer-Free Representation

So far we have relied on the separation symbol | and an integer l to label every mention with its cluster identity. Because the number of mentions in a document is quite large, we consider ways to avoid these two symbols. One way is to hard-code the cluster information directly in the mention boundaries by introducing a special symbol $\texttt{</m}_{l}\texttt{>} \in \mathcal{V}$ for each cluster number $l = 1, \ldots, C$. Under this scheme, we may assign to the annotation $x = (a, b)$ and $S = \{(1, 1, 1)(2, 2, 1)\}$:

$$\begin{aligned} y &= (\texttt{<m>}, \texttt{<c>}, \texttt{</m}_1\texttt{>}, \texttt{<m>}, \texttt{<c>}, \texttt{</m}_1\texttt{>}, \texttt{</s>}) \\ z &= (\texttt{<s>}, \texttt{<m>}, a, \texttt{</m}_1\texttt{>}, \texttt{<m>}, b, \texttt{</m}_1\texttt{>}, \texttt{</s>}) \end{aligned} \tag{7.4}$$

The performance of this representation was found to be surprisingly poor, likely because the model has a hard time learning to predict so many new symbols. We tackle this issue by introducing a "new" action `<new>` that delegates the burden of specifying an unseen cluster number to postprocessing. The example (7.4) is now assigned the action sequence

$$\begin{aligned} y &= (\texttt{<m>}, \texttt{<c>}, \texttt{<new>}, \texttt{<m>}, \texttt{<c>}, \texttt{</m}_1\texttt{>}, \texttt{</s>}) \\ z &= (\texttt{<s>}, \texttt{<m>}, a, \texttt{</m}_1\texttt{>}, \texttt{<m>}, b, \texttt{</m}_1\texttt{>}, \texttt{</s>}) \end{aligned}$$

In this way, whenever the model predicts `<new>` we can feed $\texttt{</m}_{l+1}\texttt{>}$ as the decoder input where l is the previous number of clusters. The model is only responsible for predicting $\texttt{</m}_{l}\texttt{>}$ for expanding a known cluster l. We call this **integer-free representation** and show that it is nearly as performant as a corresponding baseline that relies on $|$ and l while being shorter and notationally cleaner.

7.3.4 Mention Alignment

The issue of alignment arises only under partial linearization and token action. In this case, it is possible to have linearized mentions whose corresponding locations in the input document are ambiguous. Consider the document $x = (a, b, c, d, e, b, b)$. The linearization consisting of two length-1 mentions

$$\texttt{<m>}\ b \mid 1\ \texttt{</m>}\ \texttt{<m>}\ b \mid 1\ \texttt{</m>} \tag{7.5}$$

correspond to either $S = \{(2, 2, 1), (6, 6, 1)\}$ or $S = \{(6, 6, 1), (7, 7, 1)\}$. We align mentions by aligning tokens with gaps (i.e., a token may be aligned to nothing). Prior to aligning tokens, we remove all special symbols while saving the span information for each mention (omitting the cluster information for simplicity). For instance, (7.5) becomes (b, b) with spans $(1, 1)$ and $(2, 2)$.

We then find a highest-scoring alignment where the score measures token matching (1 if matched, -1 if mismatched) and length-n affine gap penalty $g(n) = -1 - p(n - 1)$, where p is a hyperparameter[1]. The affine gap penalty encourages long unsegmented gaps, capturing the intuition that mentions tend to appear in high density. The above example has the optimal alignment with 2 matches ($6 \leftrightarrow 1$ and $7 \leftrightarrow 2$) and a length-5 gap in the source sequence $(1 \ldots 5)$. Plugging in the token matches in the span information, we identify the locations $(6, 6)$ and $(7, 7)$ corresponding to the second annotation. This approach naturally handles nested mentions.

For a document of length T' and a partial linearization (with special symbols removed) of length K, there are $\binom{T'+K}{K}$ possible alignments with gaps. We exactly find an optimal alignment in $O(T'K)$ time by Gotoh's algorithm [173]. An oracle experiment shows that our approach is highly effective, reaching 98.0 average F1 on OntoNotes if we use the gold partial linearization. We further improve the alignment scheme by inserting sentence markers in the input document and linearization, which allows us to constrain the alignment to sentence pairs at test time.

7.4 Discussion

Having presented our technical approach, we discuss how it relates to other approaches to coreference resolution that also use sequence-based models.

Our approach is *pure* seq2seq because we use *both* the standard architecture and the system de-

[1]In our experiment, we set $p = 0$ for simplicity, as we observed negligible performance differences when $p \leq 0.0001$.

sign that is applicable to any seq2seq task, such as machine translation and text summarization. In contrast, the approach of Bohnet *et al.* [54] is not considered pure seq2seq because their transition-based system is specifically designed for the coreference task (e.g., their "Link" and "Append" actions are meant to cluster mentions together), even though the system itself is implemented using the standard seq2seq architecture. To understand the practical difference, note that their system requires M encoder forward passes for a single document during inference where M is the number of sentences, whereas ours requires one. The success of the transition system of Bohnet *et al.* [54] does not imply the success of the general seq2seq system in our work.

We adopt the generalized seq2seq framework that may require postprocessing during generation (e.g., for copy action) for improved modeling flexibility and performance, but the postprocessing step does not change the standard seq2seq architecture and system design. Furthermore, our model *without* postprocessing is nearly as effective. Specifically, our "Full linear + token action + T0_{3B}" model in Table 7.2 is a standard seq2seq model with no postprocessing during inference but achieves an 82.4 test F1 score, which is competitive with our best same-size "Full linear + copy action + T0_{3B}" model that does require token-level postprocessing (82.9).

We use constrained decoding during generation to ensure valid coreference annotation, which is a standard practice [172, 22]. The details can be found in Appendix E.1.

7.5 Experiments

7.5.1 Datasets

We train and evaluate on three widely used datasets for coreference resolution: OntoNotes [168], PreCo [169] and LitBank [171]. The data statistics are summarized in Table 7.1. It is important to note that these datasets exhibit significant variations in terms of size, document length, number of mentions/clusters, and domain. Following Kirstain *et al.* [52], we incorporate the speaker's name into the text whenever there is a change in speakers for datasets that include speaker metadata. In addition to training and evaluating on these three datasets individually, we also perform an additional experiment involving joint training on the combined dataset comprising all three datasets.

	# Docs					
Dataset	Train	Dev	Test	Words	Mentions	Cluster Size
OntoNotes	2802	343	348	467	56	4.4
LitBank	80	10	10	2105	291	3.7
PreCo	36120	500	500	337	105	1.6

Table 7.1: Data statistics for OntoNotes, LitBank, and PreCo datasets. The number of documents in each split, average word count per document, average mention count per document, and average mention count per cluster are listed.

To address the issue of data magnitude imbalance in joint training, we adopt the methodology suggested by Toshniwal *et al.* [170] and downsample the OntoNotes and PreCo datasets to 2K samples per epoch.

7.5.2 Implementation Details

We initialize our model using the T5 model family [166], which includes models of various sizes. Specifically, we use T5 [166], T0 [167], and FLAN-T5 [174] with model sizes base, large, XL/3B, and XXL/pp. We use the pretrained models available in the Hugging Face Transformers library [175].

To train large models with limited resources, we use Deepspeed [176] with ZeRO optimizers [177] and enable gradient checkpointing. We divide the document into overlapped segments, each with a maximum length of 2048 tokens and an overlapping length of 1024 tokens. During inference, the maximum input length is 4096 tokens for all our experiments. We use constrained beam search with beam size 4.

For optimization, we use the Adam optimizer [96] with the learning rate of 5e-4 for base/large models, 5e-5 for XL/3B, and 3e-5 for XXL/pp models. We use a linear learning rate decay scheduler with a warmup proportion of 0.1. We train our models using a batch size of 1 per GPU, using 8 A100 40G GPUs for models of size up to 3B and 8 A100 80G GPUs for models of size 11B.

7.5.3 Baselines

We categorize the baselines into the following three groups.

		MUC			B^3			CEAF$_{\phi_4}$			Avg.
	Model	P	R	F1	P	R	F1	P	R	F1	F1
Non-Seq2seq	Lee *et al.* [165]	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
	Lee *et al.* [50]	81.4	79.5	80.4	72.2	69.5	70.8	68.2	67.1	67.6	73.0
	Joshi *et al.* [178]	84.7	82.4	83.5	76.5	74.0	75.3	74.1	69.8	71.9	76.9
	Yu *et al.* [179]	82.7	83.3	83.0	73.8	75.6	74.7	72.2	71.0	71.6	76.4
	Joshi *et al.* [180]	85.8	84.8	85.3	78.3	77.9	78.1	76.4	74.2	75.3	79.6
	Xia *et al.* [181]	85.7	84.8	85.3	78.1	77.5	77.8	76.3	74.1	75.2	79.4
	Toshniwal *et al.* [182]	85.5	85.1	85.3	78.7	77.3	78.0	74.2	76.5	75.3	79.6
	Wu *et al.* [163]*	88.6	87.4	88.0	82.4	82.0	82.2	79.9	78.3	79.1	83.1
	Xu and Choi [51]	85.9	85.5	85.7	79.0	78.9	79.0	76.7	75.2	75.9	80.2
	Kirstain *et al.* [52]	86.5	85.1	85.8	80.3	77.9	79.1	76.8	75.4	76.1	80.3
	Dobrovolskii [183]	84.9	87.9	86.3	77.4	82.6	79.9	76.1	77.1	76.6	81.0
	Toshniwal *et al.* [170]	-	-	-	-	-	-	-	-	-	79.6
	Liu *et al.* [53] + T0$_{3B}$	85.8	88.3	86.9	79.6	83.3	81.5	78.3	78.5	78.4	82.3
	Liu *et al.* [53] + FLAN-T5$_{XXL}$	86.1	88.4	87.2	80.2	83.2	81.7	78.9	78.3	78.6	82.5
Transition Seq2seq	Bohnet *et al.* [54] + mT5$_{XXL}$	87.4	88.3	87.8	81.8	83.4	82.6	79.1	79.9	79.5	83.3
Seq2seq	Paolini *et al.* [56]+T5$_{base}$	-	-	81.0	-	-	69.0	-	-	68.4	72.8
	Paolini *et al.* [56]+T0$^{\dagger}_{3B}$	85.0	86.0	85.2	76.1	78.5	77.3	76.5	75.6	76.0	79.6
	Partial linear + T0$_{3B}$	83.9	87.6	85.7	76.6	82.1	79.3	77.7	76.5	77.1	80.7
	Integer free + T0$_{3B}$	84.9	88.8	86.8	78.9	84.0	81.4	78.1	79.3	78.7	82.3
	Full inear + token action + T0$_{3B}$	85.9	88.6	87.2	79.6	83.5	81.5	78.9	78.0	78.5	82.4
	Full linear + copy action + T0$_{3B}$	85.8	89.0	87.4	80.0	84.3	82.1	79.1	79.4	79.3	82.9
	Full linear + copy action + T0$_{pp}$	86.1	89.2	87.6	80.6	84.3	82.4	78.9	80.1	79.5	83.2

Table 7.2: Results on the OntoNotes (CoNLL-12 English) test set. The average CoNLL F1 score of MUC, B^3 and CEAF$_{\phi_4}$ is the main evaluation criterion. Models marked with † are our implementation. $*$ marks models using additional training data.

Non-seq2seq Models in this category are specifically designed for coreference resolution and employ sophisticated coreference-specific architectures. Most models in this category follow the approach introduced by Lee *et al.* [165] to detect mention spans in the input text and establish antecedent relationships by computing span representation similarity at either span level [165, 50, 178, 180, 51, 53] or word level [52, 183]. In contrast, Wu *et al.* [163] use a QA model to predict coreferent mention boundaries, while Xia *et al.* [181], Toshniwal *et al.* [182], and Toshniwal *et al.* [170] use memory augmented models.

Transition-based Seq2seq Models in this category are based on a designed transition system. Currently Bohnet *et al.* [54] is the only model in this category.

Seq2seq Models in this category leverage a sequence-to-sequence architecture to predict linearized coreference annotations without relying on coreference-specific model architectures or specialized system designs. Existing models in this category include Urbizu *et al.* [55] and Paolini *et al.* [56]. However, Urbizu *et al.* [55] do not evaluate on standard coreference resolution datasets like OntoNotes, and their performance is not competitive enough, so we do not compare with them. On the other hand, Paolini *et al.* [56] do not report performance using larger T5 models, and their input length is shorter than ours (1024 words). To ensure a fairer comparison, we implement their method and include the results by training and evaluating with a larger model (T0_{3B}) using the same input length as ours (2048 tokens). All of our models fall into this category. We include both full linearization models and partial linearization models as baselines. For full linearization models, we consider variants that use token action sequence and copy action sequence.

7.5.4 Results

7.5.4.1 English OntoNotes

Table 7.2 shows our main results on the test portion of English OntoNotes. We first point out that it is generally challenging to ensure full comparability due to numerous ways the approaches differ. For instance, CorefQA [163] achieves strong performance with a relatively small model (SpanBERT-large, 340M parameters), but it uses additional training data to optimize the mention proposal network (without which the performance drops to 75.9) and is slow at test time because it runs an extractive reader on each mention. On the other hand, the best results obtained with ASP [53] and the transition-based system [54] rely on much larger models ($\text{FLAN-T5}_{\text{XXL}}$, 11B parameters; mT5_{XXL}, 13B parameters), where running such large models is not always feasible depending on the authors' available computational resources (e.g., we do not have the resources to scale to 13B parameters). We do our best to interpret the results as objectively as possible[2].

The first observation is that models based on a sizeable T5 outperform those that are not (with

[2]While the model sizes are similar, the comparison with Bohnet *et al.* [54] is not fully equivalent due to their focus on multilingual coreference using the mT5 model.

Model	PreCo	LitBank	LitBank$_0$
Xia and Van Durme [184]	88.0	76.7*	-
Thirukovalluru *et al.* [185]	-	78.4	-
Wu and Gardner [186]	85.0	-	-
Toshniwal *et al.* [182]	-	76.5	-
Toshniwal *et al.* [170]	87.8	79.3	77.2
Toshniwal *et al.* [170]+Joint	87.6	-	78.2
Our copy action + T0$_{3B}$	88.5	78.3	77.3
Our copy action + T0$_{3B}$ + Joint	88.3	-	81.2

Table 7.3: Results on Preco, Litbank test set. The average CoNLL F1 score of MUC, B^3 and CEAF$_{\phi_4}$ is the evaluation metric. We report both the 10-fold cross-validation results (official setting) and the results of split 0 (LitBank$_0$) following Toshniwal *et al.* [170]. Joint denotes training on the union of OntoNotes, PreCo and LitBank$_0$. * marks transfer learning results which uses additional pretraining.

the exception of CorefQA which uses additional training data). Focusing on models with 3B parameters and input length 2048, our T0-based seq2seq model with full linearization and copy action achieves an average F1 score of 82.9, outperforming ASP (82.3) and TANL (79.6) using the same base model and input length. In fact, our 3B-model outperforms ASP using a 11B model (82.5). While full linearization with copy action is the most performant, full linearization with token action (which requires no postprocessing during generation) is almost as competitive (82.4). Partial linearization with token action using the alignment method in Section 7.3.4 also obtains a nontrivial F1 of 80.7, outperforming most non-seq2seq baselines. In terms of performance and speed, partial linearization demonstrates lower accuracy but faster inference compared to full linearization. On the English OntoNotes development data, the inference time for partial linearization is about 22 minutes, whereas full linearization takes approximately 40 minutes for both token action and copy action sequences.

With the 11B-parameter T0$_{pp}$ initialization, our model reaches 83.2 F1, better than CorefQA and close to 83.3 obtained by 13B-parameter transition-based system. While we do not have the computational budget to train a 13B-parameter model, as a point of comparison, Bohnet *et al.* [54] report the dev average F1 of 78.0 using a 3.7B-parameter mT5$_{XL}$; our model using the same base model with full linearization and copy action obtains 79.6.

7.5.4.2 PreCo and LitBank

We further verify the effectiveness of our seq2seq model by taking the same 3B-parameter T0 setting (full linearization, copy action) for OntoNotes to additional datasets in Table 7.3. We consider PreCo [169] and LitBank [171] following the same experimental setup in Toshniwal *et al.* [170]. On PreCo, which provides the largest training dataset (36K documents, compared to 2.8K in OntoNotes), our model outperforms all previous works at an average F1 score of 88.5.

LitBank, on the other hand, is a very small dataset with only 100 annotated documents split into 80-10-10 for training, validation, and test. Its official evaluation metric is an average over 10-fold cross-validation results; we also report results on split 0 to compare with prior work. Despite the small training data, our model achieves competitive results of 77.3 on split 0 and 78.3 on cross-validation, though lagging behind the task-specific model of Toshniwal *et al.* [170]. When the model is trained on the union of OntoNotes, Preco and LitBank split 0 training portion, the model achieves a significantly better performance of 81.2, beating 78.2 in the previous work. This shows that (1) when there is sufficient training data, our seq2seq model can easily obtain state-of-the-art results, and (2) even when that is not the case, the performance is still relatively strong and can be easily improved by including other available datasets.

7.5.5 Ablation Studies

In this section, we conduct ablation studies to investigate different aspects related to sequence representations, decoder input choices, and pretrained models of varying sizes. Unless specified, all the ablation experiments are conducted using the $T0_{3B}$ model.

7.5.5.1 Action Sequence

To analyze the impact of sequence representation, we present the results of an ablation study in Table 7.4.

	Sequence Representation	Avg. F1
full linear	Baseline	82.6
	– Copy action + integer free	82.6
	– Copy action + put integer before	82.3
	– Token action + integer	82.5
	– Token action + antecedent string	79.5
partial linear	Baseline	81.1
	– w/o sentence marker	79.9
	– Oracle align	99.2
	– Oracle align w/o sentence marker	98.0

Table 7.4: Ablation study for sequence representations on OntoNotes development set. Average CoNLL F1 is reported.

Full Linearization The baseline for full linearization utilizes copy action sequence and represents cluster identity using integers. In Section 7.3.3, we introduce an integer-free representation which achieves performance comparable to the baseline (both achieving 82.6). Although the integer-free representation is more complex in design, it offers a cleaner and shorter sequence representation. Notably, placing the integer before the mention string leads to a noticeable drop in performance (82.3 vs 82.6), emphasizing the importance of predicting cluster identity after detecting mentions due to the autoregressive nature of the seq2seq model. Additionally, replacing the copy action in the baseline with a token action results in slightly worse performance (82.5 vs 82.6), indicating that a smaller action space is beneficial. Moreover, using the antecedent mention string to link coreferent mentions (similar to TANL [56]) significantly decreases performance (79.5 vs 82.6). This demonstrates the superiority of using integers to represent cluster identity over using antecedent mention strings for linking coreferent mentions.

Partial Linearization Partial linearization is incompatible with the copy action since the model skips the gap between mentions. For partial linearization, the baseline employs a token action sequence with explicit sentence markers. Sentence markers prove to be useful for the model, allowing it to focus on each sentence individually during generation and aiding in alignment. Removing sentence markers leads to a significant deterioration in performance (79.9 vs 81.1). To

Decoder Input	Avg. F1
Baseline	82.6
– Copy action sequence	56.4
– Token sequence + copy action sequence	82.5

Table 7.5: Ablation study for decoder input on OntoNotes development set. Average CoNLL F1 is reported.

further understand the benefits of sentence markers for alignment, we conduct oracle experiments using gold linearization with and without sentence markers, obtaining average F1 scores of 99.2 and 98.0, respectively. These results validate the effectiveness of sentence markers in alignment.

7.5.5.2 Decoder Input

We present an ablation study for decoder input in Table 7.5. The baseline uses a linearized token sequence as the decoder input. Replacing the token sequence with a copy action sequence (similar to Urbizu *et al.* [55]) yields significantly worse performance compared to the baseline (56.4 vs 82.6). Averaging token and action embeddings as the input embedding is also less effective than the baseline (82.5 vs 82.6). These results emphasize the importance of providing the decoder with a linearized token sequence.

7.5.5.3 Pretrained Model

Table 7.6 shows an ablation study for pretrained model. We observe an improvement in performance as the model size increases. For models of the same size, both FLAN-T5 and T0 surpass the performance of the original T5 model. T0 achieves better performance than FLAN-T5 when compared at the same size.

7.5.6 Error Analysis

To better understand the model behavior, we conduct error analysis on the dev set in Table 7.7. The experiments are based on the T0_{3B} copy action model. The unlabeled mention detection F1 is 89.2. On the other hand, the clustering performance reaches 95.8 average F1 when restricted to

Pretrained Model	# params	Avg. F1
T5_{base}	220M	76.2
T5_{large}	770M	77.2
T5_{3B}	3B	81.6
$\text{FLAN-T5}_{\text{XL}}$	3B	82.5
T0_{3B}	3B	82.6
$\text{FLAN-T5}_{\text{XXL}}$	11B	82.9
T0_{pp}	11B	83.0

Table 7.6: Ablation study for pretrained model on OntoNotes development set. Average CoNLL F1 is reported.

	F1
Mention detection	89.2
Detected mention clustering	95.8
Oracle mention clustering	94.8

Table 7.7: Error analysis on OntoNotes development set. We report mention detection F1 and mention clustering average CoNLL F1.

correctly recovered mentions, and 94.8 when we assume perfect mention detection. This shows that mention detection is the performance bottleneck; once correct mentions are obtained, the model can accurately infer coreference clusters. Upon qualitative analysis of randomly sampled gold clusters and their best matches, we find that a major source of error is annotation mistakes. For instance, one gold annotation dictates "you do not want to $[\text{face}]_{17}$ the dilemma. But $[\text{it}]_{17}$ can not be avoided" while our model correctly predicts "you do not want to face $[\text{the dilemma}]_{20}$. But $[\text{it}]_{20}$ can not be avoided"; another gold annotation dictates "[[William $]_9$ and she $]_{10}$ saw each other, it was such a wonderful reunion for [them $]_{10}$ to just hug, and he would hug [her $]_2$ and look at [her $]_2$ " while our model predicts "[[William $]_{11}$ and [she $]_2$ $]_{12}$ saw each other, it was such a wonderful reunion for [them $]_{12}$ to just hug, and [he $]_{11}$ would hug [her $]_2$ and look at [her $]_2$".

7.6 Conclusions

We have presented a highly performant seq2seq reduction of coreference resolution. Unlike in previous works that rely on task-specific approaches to obtain strong performance, we use a standard encoder-decoder model that receives the document as an input sequence and predicts a sequence representation of its coreference annotation as the target sequence. Contrary to previously reported weak results using seq2seq reductions, we show for the first time that with suitable definitions of the sequence representation of the task, it is possible to achieve state-of-the-art performance, reaching 83.2 test average F1 on English OntoNotes with an 11B-parameter $T0_{pp}$ initialization. Our model's strong results fundamentally challenge the default task-specific approach to coreference resolution.

CHAPTER 8
CONCLUSIONS

8.1 Summary

In this thesis, we explored two key capabilities that enable modern NLP systems to better access external knowledge and reason about real-world entities: knowledge-intensive language processing and entity-centric language understanding.

Chapter 2 provides the necessary background for these two directions. We begin with a formal definition of information retrieval (IR), reviewing approaches ranging from classical term-based methods to neural models, including dense and generative retrieval. We discuss the strengths and limitations of each, motivating our focus on dense retrieval for its efficiency, scalability, flexibility, and generalizability. We introduce the widely adopted Noise Contrastive Estimation (NCE) objective for training dense retrievers and present commonly used evaluation metrics. We then cover retrieval-augmented generation (RAG), an end-to-end framework for addressing knowledge-intensive tasks. We formalize the RAG problem and review training paradigms, including pipeline training, joint training, and generator-guided retriever optimization. Evaluation metrics tailored to RAG benchmarks are also discussed. Next, we introduce the foundations of entity-centric language understanding. We present the formal problem formulation and major modeling approaches for entity linking, followed by a similar treatment of coreference resolution, covering representative methods and evaluation practices. This chapter lays the groundwork for the thesis's contributions to retrieval-based and entity-aware modeling.

In Chapter 3, we provide a theoretical analysis of the role of hard negatives in NCE training. We formalize how hard negatives influence the bias of the NCE gradient relative to the ideal full softmax (cross-entropy) gradient and show that hard negatives can effectively reduce this bias. Our theoretical findings are supported by empirical results. By jointly optimizing both the selection of hard negatives and the retriever architecture, we achieve new state-of-the-art performance on the challenging Zeshel dataset [66].

In Chapter 4, we introduce a multitask learning framework for training retrieval models across diverse tasks. While multitask retrieval offers practical advantages, it often underperforms compared to task-specific models. We demonstrate that encouraging task specialization within a multitask framework can substantially close this gap. Our approach combines task-specific prompting on a multitask-pretrained model with an adaptive learning strategy guided by per-parameter task sensitivity. This method leads to strong performance on the KILT retrieval benchmark.

In Chapter 5, we introduce ImpRAG, a query-free retrieval-augmented generation (RAG) system that captures information needs implicitly, without relying on human-specified queries. We unify retrieval and generation within a single decoder-only language model by partitioning its layers into specialized groups and jointly optimizing both components. By sharing the same model parameters and forward pass across retrieval and generation, ImpRAG minimizes the retriever-generator mismatch common in traditional RAG systems. ImpRAG achieves strong performance across eight knowledge-intensive tasks, outperforming standard RAG baselines and demonstrating robust generalization to unseen tasks with diverse formats.

In Chapter 6, we propose EntQA, a new approach to entity linking that reframes the problem as an inverse open-domain question answering task. This design addresses the challenge of predicting mentions without knowing the corresponding entities. EntQA first retrieves a variable number of candidate entities using a multi-label retriever trained with our novel multi-label NCE objective, then predicts potentially multiple mentions for each retrieved entity. Our simple yet effective pipeline leverages recent advances in dense retrieval and reading comprehension. EntQA achieves new state-of-the-art results on the GERBIL entity linking benchmark, without relying on KB-specific mention dictionaries or model-specific pretraining.

In Chapter 7, we present a highly effective yet remarkably simple sequence-to-sequence formulation of coreference resolution. In contrast to prior work that relies on specialized architectures and extensive hyperparameter tuning, we use a standard encoder-decoder model that takes the document as input and outputs a sequence encoding its coreference structure. We demonstrate, for the first time, that with an appropriate design of the output representation, such a straightforward approach can match or exceed the performance of task-specific models on standard coreference resolution benchmarks.

8.2 Future Directions

This thesis opens several promising directions for future research:

Improving Evaluation for Knowledge-Intensive Language Processing. In this thesis, we primarily adopted the widely used KILT benchmark [6] and simple string-matching metrics for evaluation. However, with the rapid progress of large language models, existing short-form benchmarks are becoming insufficient to reflect real-world reasoning and knowledge integration capabilities. Moreover, exact-match-based metrics are increasingly unreliable, particularly in settings where multiple correct answers exist or where semantic correctness matters more than surface form [187]. Future work should focus on (1) constructing larger and more realistic benchmarks that better reflect real-world use cases and (2) developing robust, semantically grounded evaluation metrics that provide a more accurate measure of model performance in knowledge-intensive tasks.

Adaptive Retrieval-Augmented Generation. In this thesis, we primarily focused on knowledge-intensive tasks where retrieval is triggered only once given a query or instruction. However, in real-world scenarios, retrieval and generation are often interleaved, and retrieval may need to be triggered multiple times adaptively. Prior work has explored dynamic retrieval using supervised fine-tuning on LLM-generated trajectories [44] or reinforcement learning [45]. While these methods achieve performance gains, they are typically trained and evaluated in highly synthetic settings [126, 188]. Moreover, supervised fine-tuning relies heavily on strong LLMs to produce high-quality trajectories, which often amounts to distillation rather than true capability enhancement. Similarly, RL-based approaches primarily demonstrate gains compared to prompting-only or basic SFT baselines, making it unclear whether genuine adaptive retrieval capabilities are learned.

Future work should (1) evaluate adaptive RAG under more realistic and diverse settings beyond simple benchmarks like Yang *et al.* [126], (2) reduce dependence on LLM distillation, and (3) compare against strong RAG instruction-tuned baselines (e.g., RA-IT from Chapter 5) to obtain trustworthy and meaningful conclusions. Building on these insights, research can focus on developing principled methods to improve adaptive retrieval-augmented generation.

Latent Retrieval-Augmented Generation. In Chapter 5, we demonstrated that LLMs can perform latent retrieval by leveraging their hidden states for document retrieval. However, our current approach still integrates the full retrieved texts into the generation process, which remains computationally expensive in both speed and memory. A promising future direction is to explore fully latent RAG, where both retrieval and the integration of retrieved information occur entirely in the embedding space. Such an approach could (1) improve efficiency by reducing the reliance on raw document text, (2) achieve better alignment between retrieval and generation through a shared latent space, and (3) naturally enable multi-modal retrieval-augmented generation, since embeddings from different modalities can be unified in the same representational space.

Appendices

APPENDIX A
APPENDIX TO CHAPTER 3

A.1 Percentage of Hard Negatives

We show top-64 validation recalls with varying values of the hard negative percentage p in training below:

Mixed-p (%)	DUAL	MULTI-8
0 (Random)	91.08	91.13
25	92.18	92.74
50	91.75	92.76
75	92.24	93.41
100 (Hard)	92.05	93.27

The presence of hard negatives is clearly helpful, but the exact choice of $p > 0$ is not as important. We choose $p = 50$ because we find that the presence of some random negatives often gives slight yet consistent improvement.

A.2 Reranking Experiments

We show the normalized and unnormalized accuracy of a reranker as we change the architecture while holding the retriever fixed:

Model	Normalized		Unnormalized	
	Val	Test	Val	Test
DUAL	60.43	62.49	54.87	54.73
POLY-16	60.37	60.98	54.82	53.37
POLY-64	60.80	61.88	55.20	54.15
POLY-128	60.60	62.72	55.03	54.92
MULTI-8	61.56	62.65	55.90	54.87
MULTI-64	61.94	62.94	56.23	55.15
MULTI-128	61.67	62.95	55.98	55.17
SOM	65.38	65.24	59.35	57.04
GENPOLY-128	65.89	64.98	59.82	56.82
JOINT	**76.17**	**74.90**	**69.14**	**65.42**
Logeswaran *et al.*	76.06	75.06	–	55.08
Wu *et al.*	78.24	76.58	–	–
JOINT (ours)	78.82	77.09	58.77	56.56

GENPOLY-m denotes a generalized version of the poly-encoder in which we use m leftmost entity embeddings rather than one: $s_\theta(x, y) = 1_m^\top F_{1:m}(y)^\top C_m(x, y)$. We use a trained dual encoder with 91.93% and 83.48% validation/test recalls as a fixed retriever. The accuracy increases with the complexity of the reranker. The dual encoder and the poly-encoder are comparable, but the multi-vector, the sum-of-max, and the generalized poly-encoder achieve substantially higher accuracies. Not surprisingly, the joint encoder achieves the best performance. We additionally show reranking results using the BM25 candidates provided in the Zeshel dataset for comparison with existing results. Our implementation of JOINT with BERT-base obtains comparable accuracies.

A.3 Bias Experiments on Zeshel

We consider the dual encoder $s_\theta(x, y) = E_1(x)^\top F_1(y)$ where E and F are parameterized by BERT-bases. We randomly sample 64 mentions, yielding a total of 128 entities: 64 referenced by the mentions, and 64 whose descriptions contain these mentions. We consider these 128 entities to constitute the entirety of the label space $\mathcal{Y}$. On the 64 mentions, we estimate $J_{\text{CE}}(\theta)$ by normalizing over the 128 entities; we estimate $J_{\text{HARD}}(\theta)$ by normalizing over $K = 8$ candidates where 7 are drawn from a negative distribution: either random, hard, or mixed. Instead of a single-sample estimate as in (3.11), we draw negative examples 500 times and average the result. We estimate the bias $b(\theta) \in \mathbb{R}^d$ by taking a difference between these two estimates and report the norm below:

Negatives	$\|\|b(\theta_{\text{CE}})\|\|$	$\|\|b(\theta_{\text{RAND}})\|\|$
Random	16.33	166.38
Hard	0.68	0.09
Mixed-50	1.20	0.90

We consider two parameter locations. θ_{CE} is obtained by minimizing the cross-entropy loss (92.19% accuracy). θ_{RAND} is obtained by NCE with random negatives (60% accuracy). The bias is drastically smaller when negative examples are drawn from the model instead of randomly. Mixed negatives yield comparably small biases. With random negatives, the bias is much larger at θ_{RAND} since $\nabla J_{\text{CE}}(\theta_{\text{RAND}})$ is large. In contrast, hard and mixed negatives again yield small biases.

APPENDIX B
APPENDIX TO CHAPTER 4

B.1 Algorithm in Matrix Form

Alogrithm 1 is the matrix form of our adaptive learning algorithm.

Algorithm 1 Task sensitivity-guided adaptive learning

```
Require: Model parameter $\theta \in \mathbb{R}^d$; minibatches $\mathcal{B}$ where each batch $B \in \mathcal{B}$ is further divided by tasks $B = \{B_k\}_{k=1...K}$; moving average rate $\beta \in [0,1]$; temperature $\tau > 0$; learning rate $\eta > 0$
Ensure: $\mathrm{median} : \mathbb{R}^{d\times K} \to \mathbb{R}^K$ is the column-wise median; $\mathrm{softmax} : \mathbb{R}^{d\times K} \to \mathbb{R}^{d\times K}$ is the row-wise softmax; $\mathbf{1}_K$ is a vector of $K$ ones; $\odot$ is the Hadamard product.
1: Initialize $I \leftarrow \mathbf{0} \in \mathbb{R}^{d\times K}$.
2: for each batch $B = \{B_k\}_{k=1...K}$ in $\mathcal{B}$ do
3:     Compute the task-specific loss $J_k(\theta)$ on $B_k$ for each $k = 1 \dots K$.
4:     Compute the gradient matrix $G \in \mathbb{R}^{d\times K}$ with each column $G_k \leftarrow \nabla J_k(\theta)$.
5:     Compute the sensitivity matrix $I' \in \mathbb{R}^{d\times K}$ with each column $I'_k \leftarrow G_k \odot \theta$.
6:     Normalize the sensitivity scales across tasks $I' \leftarrow I'\mathrm{diag}\left(\mathrm{median}(I')\right)^{-1}$.
7:     Update the moving average $I \leftarrow \beta I + (1-\beta)I'$.
8:     Update the parameter $\theta \leftarrow \theta - \eta(G \odot U)\mathbf{1}_K$ where $U \leftarrow \mathrm{softmax}(I/\tau) \in \mathbb{R}^{d\times K}$.
9: end for
```

B.2 Data Details

See Table B.1 for data statistics and some data-related hyperparameters. We randomly downsample T-REx and zsRE to bring them to the same order of magnitude as the others. We follow [2] and use temperature-scaled mixing sampling strategy to compute batch size for each task k: $B_k \propto (N_k / \sum_{k'=1}^{K} N_{k'})^{1/c}$ for some temperature c (we set it to 4 in our experiments). Here N_k is the dataset size of task k. Note that we compute task loss of each task batch independently instead of mixing all task batches for every optimization step. Each dataset needs to sample different number of batches to cover every training sample in that dataset once. We set the maximum of them as the number of batches that every dataset needs to sample. We shuffle and cycle batch

Dataset	#Train	B	L
Natural Questions	76k	16	32
TriviaQA	53k	14	32
HotpotQA	69k	15	32
Wizard of Wikipedia	80k	16	256
T-REx	95k	16	32
FEVER	71k	15	64
Zero Shot RE	100k	17	32
AIDA-YAGO 2	18k	11	128

Table B.1: Data statistics and some data-related hyperparameters for our experiments. B denotes batch size. L denotes query maximum input length excluding the task prefix.

sampling iterators of datasets that finish iterating early. Batch size of each dataset computed by setting mixing temperature $c = 4$ and $\sum_{k'=1}^{K} N_{k'} = 120$ is in Table B.1.

B.3 Other Training Details

lr	warmup	#negs	epochs	τ	β	B_{total}
5e-6	0.1	2	3	2	0.999	120

Table B.2: Training hyperparameters for training our TACO-DR model. We use Adam[96] with learning rate $5e-6$. We use linear learning rate schedule with warmup raio 0.1. Each query uses 2 hard negatives for training. Each ANCE episode trains for 3 epochs. Total batch size of all task batches are 120.

The data-related hyperparameters, such as maximum input query length and batch size, are listed in Table B.1. The training hyperparameters are listed in Table B.2. We use NCE loss with cross device in-batch negative mixed with hard negatives to compute each task loss. We sample two hard negatives for each query. We employ a "burn in" period for the first 10% training steps with uniform learning rates for parameters to declare their tendency during adaptive learning. All of our experiments are run on a machine with 8 A100-80GB GPUS. Our implementations are built upon OpenMatch [189].

B.4 Softmax Temperature and Momentum Ratio

τ	0.1	1	2	5	10	100
Avg R-prec	72.45	73.23	73.74	73.72	73.48	72.85

Table B.3: Average page-level R-precision w.r.t softmax temperature for our adaptive learning

β	0	0.6	0.7	0.8	0.9	0.999
Avg R-prec	72.91	72.98	73.02	73.16	73.61	73.74

Table B.4: Average page-level R-precision w.r.t momentum factor for our adaptive learning

Table B.3 shows the impact of softmax temperature on validation R-precision for our adaptive learning. Table B.4 shows the impact of momentum factor on validation R-precision for our adaptive learning.

B.5 Passage-level Performance

Table B.5 shows the passage-level R-precision on KILT validation data. We also list the passage-level performance from Maillard *et al.* [81] for comparison.

	Fact Check.	Slot Filling		Open Domain QA			Dial.	
Model	**FEV**	**T-REx**	**zsRE**	**NQ**	**HoPo**	**TQA**	**WoW**	**Avg**
MT-DPR*	<u>46.96</u>	53.54	41.70	28.80	38.42	24.56	24.07	36.86
Task-specific DPR*	43.92	58.54	78.81	28.13	<u>43.47</u>	23.79	20.73	42.48
Task-specific (ours)	44.89	<u>72.09</u>	**84.47**	**33.14**	43.40	**29.57**	<u>27.64</u>	<u>47.89</u>
TACO	**60.76**	**72.57**	<u>82.80</u>	<u>31.16</u>	**46.72**	<u>28.32</u>	**33.24**	**50.80**

Table B.5: Passage-level R-precision on KILT validation data. **Bold** indicates the best model and <u>underline</u> indicates the second. $*$ marks results from Maillard *et al.* [81].Only page-level R-precision is defined for AIDA.

APPENDIX C
APPENDIX TO CHAPTER 5

C.1 Passage Encoding

Given k retrieved passages, we must obtain the key-value (KV) states from the middle layer group $\mathcal{L}_M$ to enable cross-attention. We explore three passage encoding strategies, summarized in Table C.1.

First, we consider Independent Encoding, where each passage is encoded separately using position IDs starting from zero, following the parallel encoding strategy in Yen *et al.* [190]. The resulting KV states are then concatenated across passages.

Second, we examine Concatenated Encoding (Segmented), in which passages are concatenated into a single sequence, but attention across passages is blocked to prevent inter-passage interaction.

Third, we evaluate Concatenated Encoding (Full Attention), where passages are concatenated and full cross-passage attention is allowed throughout the encoding.

We conduct these experiments by finetuning Llama-3.1 8B model on the Natural Questions (NQ) dataset using the top-10 passages retrieved by Contriever-MSMARCO, and report Exact Match (EM) scores on the development set. As shown in Table C.1, the two simpler strategies—Independent Encoding and Segmented Concatenation—perform similarly, while Full Attention Concatenation yields a clear performance improvement, highlighting the benefit of modeling inter-passage dependencies.

Encoding Method	Dev EM
Independent Encoding	51.7
Segmented Concatenation	51.4
Full Attention Concatenation	53.3

Table C.1: Performance of different passage encoding strategies.

C.2 Freezing Passage Representations

We investigate the impact of freezing passage representations—either hidden states or key-value (KV) states—during inference with a fixed retriever. All experiments are conducted using a fine-tuned LLaMA-3.1 8B model and the top-10 passages retrieved by Contriever-MSMARCO on the Natural Questions (NQ) dataset. Results are reported in Table C.2.

We explore two freezing strategies, both using the Independent Encoding approach described in Appendix C.1. In the first variant, Frozen Hidden States, we freeze the hidden representations of retrieved passages as produced by the initial (untrained) LLaMA-3.1 8B model, and pass them through the trained key/value projection layers to generate the KV states used in cross-attention.

In the second variant, Frozen KV States, we directly freeze the key and value attention states of the passages, also obtained from the initial LLama-3.1 8B model.

We observe that both freezing methods yield comparable performance, slightly underperforming the fully dynamic setting where passage KV states are computed using the trained model.

Method	Dev EM
No Freezing	51.4
Frozen Hidden States	50.8
Frozen KV States	50.7

Table C.2: Performance of freezing different passage representations on NQ dev set with top-10 Contriever-MSMARCO retrieved passages.

C.3 Passage KV States Compression

When we use independent encoding strategy in Appendix C.1, one benefit will be that we can save the middle layer group key value states for all the passages in knowledge corpus in disk and during inference after retrieval we can load the key value states from disk without recomputation. However, this will result in a large amount of disk spaces. Thus, we consider two compression strategies: token compression and product quantization and we conduct experiments following the same setting as the Frozen KV states in Appendix C.2. Specifically, take for token compression, we use the Heavy Hitter [191] and only keep half number of tokens for each passage. For production quantization, we use FAISS codec with index type OPQ32x128-PQ32x8 for each key value head, which is trained on 500k randomly sampled wikipedia passages. The compression rate with this quantization is $\frac{128\times2}{32} = 8$ for original bfloat16 state vector of each attention head. We report the results in Table C.3. We can see that both strategies don't hurt the performance much.

Compression	Dev EM
No Compression	50.7
Heavy Hitter	49.9
Product Quantization	50.3

Table C.3: The results for various compression techniques.

C.4 Instruction-Tuning Datasets

We use OpenAssistant Conversations Dataset (oasst1; [192]), Conversational Question Answering (CoQA; [193]), Discrete Reasoning Over Paragraphs (DROP; [194]), NewsQA [195], PubMedQA [196], QA for Artificial Intelligence (Quail; [197]), SQuAD v2 [198],[1] and CNN DailyMail [199] The templates for these datasets are shown in Table C.4.

Task	Template
Instruction-Tuning Tasks	
oasst1	{turn$_1$} {turn$_2$} {turn$_3$} ...
CoQA, DROP, NewsQA, PubMedQA, SQuAD	{context} Q: {question} A: {answer}
CNN DailyMail	{context} Summarize this article: {summary}

Table C.4: Prompt templates. We only use retrieval for knowledge-intensive tasks.

C.5 Baselines and Computational Resources

Discussions on Baselines. For all these baselines, we use the retrieve-then-generate paradigm, i.e., begin by retrieving candidates using the retrievers and then incorporate them into the context for training and inference. This implies that these baselines require an additional retriever, leading

[1]We only use answerable questions from SQuAD v2.

to increased computational costs and a higher number of model parameters compared to ImpRAG. However, since this is a standard practice for retrieval-augmented models, we continue to use them in the baselines to establish stronger comparisons.

Computational Resources. We use NVIDIA H100 GPUs. Each training session requires 8 H100 GPUs, and hosting the index also demands an additional 8 GPUs. Training the baseline approaches takes roughly 96 GPU hours, whereas our models require approximately 160 GPU hours.

APPENDIX D

APPENDIX TO CHAPTER 6

D.1 Threshold Optimization

To see if it is possible to improve over the static threshold value $\gamma = 0.05$, we tried automatically calibrating γ based on the AIDA validation performance by considering every effective threshold obtained from a sorted list of probabilities of labeled mentions. The best threshold was $\gamma = 0.03146$. The validation F1 score improved from 87.32 to 87.75, and the GERBIL test score improved from 60.46 to 60.55. Thus threshold optimization can yield a minor improvement, but overall we find that EntQA is robust to choices of threshold in a reasonable range.

D.2 Document-Level Information

We explored various ways of injecting document-level information in paragraphs. We tried the first token, the first sentence, and a continuous topic embedding (obtained by averaging all token embeddings in the document). We settled on the first-token version because it gave the best performance. For many of the GERBIL datasets, however, we obtain almost the same performance with or without the topic information. As it is somewhat dataset-specific (e.g., the first word in AIDA is always the topic word), we leave it as an option in our model for the user to decide. Table D.1 shows the GERBIL performance without any topic information vs with the first token in the document.

Table D.1: GERBIL test scores with and without using the first document token as document-level topical information.

Topic Info	**AIDA**	**MSNBC**	**Der**	**K50**	**R128**	**R500**	**OKE15**	**OKE16**	**Avg**
None	81.7	**72.2**	52.5	64.2	54.0	40.3	59.0	48.9	59.1
First token	**85.8**	72.1	**52.9**	**64.5**	**54.1**	**41.9**	**61.1**	**51.3**	**60.5**

APPENDIX E
APPENDIX TO CHAPTER 7

E.1 Constrained Decoding

We implement constrained decoding through vocabulary masking, which depends on the current state of the generated sequence. Below, we outline the masking rules for various models. For more details, please check our code.

E.1.1 Full Linearization

We categorize the state of the generated tokens thus far in full linearization with integer cluster identity as follows:

1. Inside Cluster Identity: Cluster identity generation stage (i.e., the count of mention end tokens `</m>` is less than that of separation token |).
2. Inside Mention: Open mentions exist (i.e., the count of separation tokens | is less than that of mention start tokens `<m>`), but not in Inside Cluster Identity state.
3. Outside: Not in Inside Mention or Inside Cluster Identity states.

Token Action. Depending on the current state, different tokens are permitted:

- Outside: The next token from the input source and the mention start token `<m>` are allowed.
- Inside Mention: The next token from the input source, the mention start token `<m>`, and the separation token | are allowed.
- Inside Cluster Identity: All integer tokens and the mention end token `</m>` are allowed.

Copy Action. For the Copy Action model, trained to generate a copy action token `<c>` rather than the actual next source token, we manipulate the logits scores to enforce the generation of the actual source token. Specifically, the logits score of the actual next source token is set to that of the copy token `<c>`. The copy token `<c>` is then masked out. The rules for permissible tokens remain consistent with those for Token Action.

E.1.2 Partial Linearization

Due to alignment issues with the input source in partial linearization, it's infeasible to impose constraints on source input token generation as in full linearization. Nonetheless, sentence-level constraints can be applied by utilizing sentence boundary markers `<sentence>` and `</sentence>` in both the input source and linearization. We identify the current state of the generated tokens in partial linearization based on sentence markers and integer cluster identity as:

1. Complete Sentence: Equal counts of sentence start `<sentence>` and end markers `</sentence>`.
2. Inside Sentence i: Within the i-th sentence (i.e., `<sentence>` count is i and not in Complete Sentence state).
3. Inside Cluster Identity: Cluster identity generation stage (i.e., the count of mention end tokens `</m>` is less than that of separation token $|$).
4. Inside Mention: Open mentions exist (i.e., the count of separation tokens `|` is less than that of mention start tokens `<m>`), but not in Inside Cluster Identity state.
5. Outside: Not in Inside Mention or Inside Cluster Identity states.

Depending on the current state, different tokens are permitted:

- Complete Sentence: only sentence start marker token `<sentence>` is allowed.
- Outside and Inside sentence i: All the tokens from the i-th sentence in the input source, the mention start token `<m>` and the sentence end marker token `</sentence>` are allowed.

- Inside Mention and Inside Sentence i: All the tokens from the i-th sentence in the input source, the mention start token `<m>`, the separation token | and the sentence end marker token `</sentence>` are allowed.

- Inside Cluster Identity and Inside Sentence i: All the tokens from the i-th sentence in the input source, all integer tokens, the mention end token `</m>` and the sentence end marker token `</sentence>` are allowed.

E.1.3 Integer-Free Representation.

In the integer-free model, which is trained to predict `<new>` for unseen clusters instead of $\texttt{</m}_{l+1}\texttt{>}$, where $\texttt{</m}_0\texttt{>}, \texttt{</m}_1\texttt{>}, \ldots, \texttt{</m}_l\texttt{>}$ have already appeared, we manipulate the logits scores to enforce the generation of the $\texttt{</m}_{l+1}\texttt{>}$ token instead of `<new>` token for unseen cluster. Specifically, the logits score of the $\texttt{</m}_{l+1}\texttt{>}$ token is set to that of the `<new>` token. The `<new>` token is then masked out. We categorize the state of the generated tokens thus far in full linearization with integer-free cluster identity as follows:

1. Inside Mention Seen l: Open mentions exist (i.e., the count of cluster identity hard-coded mention end tokens $\texttt{</m}_l\texttt{>}$ is less than that of mention start tokens `<m>`) and
$\texttt{</m}_0\texttt{>}, \texttt{</m}_1\texttt{>}, \ldots, \texttt{</m}_l\texttt{>}$ has been seen so far.

2. Outside Mention: No open mentions.

Depending on the current state, different tokens are permitted:

- Inside Mention Seen l: The next token from the input source, the mention start token `<m>`, and cluster identity hard-coded mention end tokens $\texttt{</m}_0\texttt{>}, \texttt{</m}_1\texttt{>}, \ldots, \texttt{</m}_l\texttt{>}, \texttt{</m}_{l+1}\texttt{>}$ are allowed.

- Outside Mention: The next token from the input source and the mention start token `<m>` are allowed.

ACKNOWLEDGMENT OF PREVIOUS PUBLICATIONS

Portions of this dissertation are based on the following first-authored publications:

- Wenzheng Zhang and Karl Stratos. “Understanding Hard Negatives in Noise Contrastive Estimation.” *Proceedings of NAACL*, 2021.
- Wenzheng Zhang, Wenyue Hua, and Karl Stratos. “EntQA: Entity Linking as Question Answering.” *International Conference on Learning Representations (ICLR)*, 2022.
- Wenzheng Zhang, Chenyan Xiong, Karl Stratos, and Arnold Overwijk. “Improving Multi-task Retrieval by Promoting Task Specialization.” *Transactions of the Association for Computational Linguistics (TACL)*, 2023.
- Wenzheng Zhang, Sam Wiseman, and Karl Stratos. “Seq2seq is All You Need for Coreference Resolution.” *Proceedings of EMNLP*, 2023.
- Wenzheng Zhang, Xi Victoria Lin, Karl Stratos, Wen-tau Yih, and Mingda Chen. “ImpRAG: Retrieval-Augmented Generation with Implicit Queries.” *arXiv preprint arXiv:2506.02279*, 2025.

These works have been incorporated into Chapters 3, 5, 6, 7, and 8 of this dissertation, respectively. I was primarily responsible for the implementation, experimentation, and analysis in each of these projects. Research design and writing were carried out collaboratively with my co-authors. All chapters have been adapted and revised to ensure cohesion and consistency within the dissertation.

REFERENCES

[1] 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.

[2] C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, 2019.

[3] T. Brown *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] J. Achiam *et al.*, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.

[5] A. Grattafiori *et al.*, “The llama 3 herd of models,” *arXiv preprint arXiv:2407.21783*, 2024.

[6] F. Petroni *et al.*, “Kilt: A benchmark for knowledge intensive language tasks,” in *NAACL-HLT*, 2021.

[7] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[8] N. Kolitsas, O.-E. Ganea, and T. Hofmann, “End-to-end neural entity linking,” in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, 2018, pp. 519–529.

[9] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, “End-to-end neural coreference resolution,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 188–197.

[10] D. Chen, A. Fisch, J. Weston, and A. Bordes, “Reading Wikipedia to answer open-domain questions,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, R. Barzilay and M.-Y. Kan, Eds., Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1870–1879.

[11] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, “FEVER: A large-scale dataset for fact extraction and VERification,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 809–819.

[12] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[13] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[14] V. Karpukhin *et al.*, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781.

[15] D. Gillick *et al.*, "Learning dense representations for entity retrieval," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 528–537.

[16] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992.

[17] K. Lee, M.-W. Chang, and K. Toutanova, "Latent retrieval for weakly supervised open domain question answering," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 6086–6096.

[18] D. Gillick, A. Presta, and G. S. Tomar, "End-to-end retrieval in continuous space," *arXiv preprint arXiv:1811.08008*, 2018.

[19] M. Douze *et al.*, "The faiss library," *arXiv preprint arXiv:2401.08281*, 2024.

[20] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over BERT," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, ACM, 2020, pp. 39–48.

[21] S. Humeau, K. Shuster, M.-A. Lachaux, and J. Weston, "Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring," in *International Conference on Learning Representations*, 2020.

[22] N. De Cao, G. Izacard, S. Riedel, and F. Petroni, "Autoregressive entity retrieval," in *International Conference on Learning Representations*, 2021.

[23] Y. Tay *et al.*, "Transformer memory as a differentiable search index," *Advances in Neural Information Processing Systems*, vol. 35, pp. 21 831–21 843, 2022.

[24] Y. Wang *et al.*, “A neural corpus indexer for document retrieval,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 600–25 614, 2022.

[25] M. Bevilacqua, G. Ottaviano, P. Lewis, W.-t. Yih, S. Riedel, and F. Petroni, “Autoregressive search engines: Generating substrings as document identifiers,” *arXiv preprint arXiv:2204.10628*, 2022.

[26] W. Sun *et al.*, “Learning to tokenize for generative retrieval,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 46 345–46 361, 2023.

[27] R. Pradeep *et al.*, “How does generative retrieval scale to millions of passages?” In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 1305–1321.

[28] M. U. Gutmann and A. Hyvärinen, “Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics,” *The journal of machine learning research*, vol. 13, no. 1, pp. 307–361, 2012.

[29] A. Mnih and Y. W. Teh, “A fast and simple algorithm for training neural probabilistic language models,” in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012, pp. 419–426.

[30] Z. Ma and M. Collins, “Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 3698–3707.

[31] V. Karpukhin *et al.*, “Dense passage retrieval for open-domain question answering,” *arXiv preprint arXiv:2004.04906*, 2020.

[32] L. Xiong *et al.*, “Approximate nearest neighbor negative contrastive learning for dense text retrieval,” in *International Conference on Learning Representations*, 2021.

[33] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, “Retrieval augmented language model pre-training,” in *International conference on machine learning*, PMLR, 2020, pp. 3929–3938.

[34] G. Izacard and E. Grave, “Leveraging passage retrieval with generative models for open domain question answering,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, P. Merlo, J. Tiedemann, and R. Tsarfaty, Eds., Online: Association for Computational Linguistics, Apr. 2021, pp. 874–880.

[35] M. Lewis *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7871–7880.

[36] S. Min, J. Michael, H. Hajishirzi, and L. Zettlemoyer, "AmbigQA: Answering ambiguous open-domain questions," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 5783–5797.

[37] G. Izacard and E. Grave, "Distilling knowledge from reader to retriever for question answering," in *International Conference on Learning Representations*, 2021.

[38] X. V. Lin *et al.*, "RA-DIT: Retrieval-augmented dual instruction tuning," in *The Twelfth International Conference on Learning Representations*, 2024.

[39] G. Izacard *et al.*, "Atlas: Few-shot learning with retrieval augmented language models," *Journal of Machine Learning Research*, vol. 24, no. 251, pp. 1–43, 2023.

[40] W. Shi *et al.*, "Replug: Retrieval-augmented black-box language models," in *NAACL-HLT*, 2024.

[41] S. Yao *et al.*, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[42] O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis, "Measuring and narrowing the compositionality gap in language models," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 5687–5711.

[43] O. Khattab *et al.*, "Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp," *arXiv preprint arXiv:2212.14024*, 2022.

[44] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," in *International Conference on Learning Representations (ICLR)*, 2024.

[45] M. Chen *et al.*, "Learning to reason with search for llms via reinforcement learning," *arXiv preprint arXiv:2503.19470*, 2025.

[46] N. Gupta, S. Singh, and D. Roth, "Entity linking via joint encoding of types, descriptions, and context," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2681–2690.

[47] J. Hoffart *et al.*, "Robust disambiguation of named entities in text," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 782–792.

[48] X. Ling, S. Singh, and D. S. Weld, “Design challenges for entity linking,” *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 315–328, 2015.

[49] J. M. van Hulst, F. Hasibi, K. Dercksen, K. Balog, and A. P. de Vries, “Rel: An entity linker standing on the shoulders of giants,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 2197–2200.

[50] K. Lee, L. He, and L. Zettlemoyer, “Higher-order coreference resolution with coarse-to-fine inference,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 687–692.

[51] L. Xu and J. D. Choi, “Revealing the myth of higher-order inference in coreference resolution,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 8527–8533.

[52] Y. Kirstain, O. Ram, and O. Levy, “Coreference resolution without span representations,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 14–19.

[53] T. Liu, Y. E. Jiang, N. Monath, R. Cotterell, and M. Sachan, “Autoregressive structured prediction with language models,” in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 993–1005.

[54] B. Bohnet, C. Alberti, and M. Collins, “Coreference Resolution through a seq2seq Transition-Based System,” *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 212–226, Mar. 2023.

[55] G. Urbizu, A. Soraluze, and O. Arregi, “Sequence to sequence coreference resolution,” in *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*, Barcelona, Spain (online): Association for Computational Linguistics, Dec. 2020, pp. 39–46.

[56] G. Paolini *et al.*, “Structured prediction as translation between augmented natural languages,” in *International Conference on Learning Representations*, 2021.

[57] M. Vilain, J. D. Burger, J. Aberdeen, D. Connolly, and L. Hirschman, “A model-theoretic coreference scoring scheme,” in *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*, 1995.

[58] A. Bagga and B. Baldwin, “Algorithms for scoring coreference chains,” in *The first international conference on language resources and evaluation workshop on linguistics coreference*, vol. 1, 1998, pp. 563–566.

[59] X. Luo, “On coreference resolution performance metrics,” in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, R. Mooney, C. Brew, L.-F. Chien, and K. Kirchhoff, Eds., Vancouver, British Columbia, Canada: Association for Computational Linguistics, Oct. 2005, pp. 25–32.

[60] W. Zhang and K. Stratos, “Understanding hard negatives in noise contrastive estimation,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 1090–1101.

[61] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, “Learning deep structured semantic models for web search using clickthrough data,” in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 2333–2338.

[62] T. Févry, N. FitzGerald, L. B. Soares, and T. Kwiatkowski, “Empirical evaluation of pretraining strategies for supervised entity linking,” in *Automated Knowledge Base Construction*, 2020.

[63] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, “What makes for good views for contrastive learning,” *arXiv preprint arXiv:2005.10243*, 2020.

[64] Y. Bengio and J.-S. Senécal, “Adaptive importance sampling to accelerate training of a neural probabilistic language model,” *IEEE Transactions on Neural Networks*, vol. 19, no. 4, pp. 713–722, 2008.

[65] Y. Luan, J. Eisenstein, K. Toutanova, and M. Collins, “Sparse, dense, and attentional representations for text retrieval,” *arXiv preprint arXiv:2005.00181*, 2020.

[66] L. Logeswaran, M.-W. Chang, K. Lee, K. Toutanova, J. Devlin, and H. Lee, “Zero-shot entity linking by reading entity descriptions,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3449–3460.

[67] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer, “Scalable zero-shot entity linking with dense entity retrieval,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6397–6407.

[68] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.

[69] K. Stratos, *Noise contrastive estimation*, http://karlstratos.com/notes/nce.pdf, Unpublished technical note. Accessed: April 8, 2021, 2019.

[70] G. Blanc and S. Rendle, "Adaptive sampled softmax with kernel based sampling," in *International Conference on Machine Learning*, 2018, pp. 590–599.

[71] A. S. Rawat, J. Chen, F. X. X. Yu, A. T. Suresh, and S. Kumar, "Sampled softmax with random fourier features," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 857–13 867.

[72] R. D. Hjelm *et al.*, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2019.

[73] 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.

[74] J. D. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, "Contrastive learning with hard negative samples," in *International Conference on Learning Representations*, 2021.

[75] 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)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[76] N. Gupta, S. Singh, and D. Roth, "Entity linking via joint encoding of types, descriptions, and context," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 2681–2690.

[77] K. Lee, M.-W. Chang, and K. Toutanova, "Latent retrieval for weakly supervised open domain question answering," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 6086–6096.

[78] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Realm: Retrieval-augmented language model pre-training," *arXiv preprint arXiv:2002.08909*, 2020.

[79] O.-E. Ganea and T. Hofmann, "Deep joint entity disambiguation with local neural attention," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 2619–2629.

[80] W. Zhang, C. Xiong, K. Stratos, and A. Overwijk, "Improving multitask retrieval by promoting task specialization," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1201–1212, 2023.

[81] J. Maillard *et al.*, "Multi-task retrieval for knowledge-intensive tasks," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 1098–1111.

[82] L. Xiong *et al.*, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," in *International Conference on Learning Representations*, 2021.

[83] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

[84] Z. Wang, Y. Tsvetkov, O. Firat, and Y. Cao, "Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models," *arXiv preprint arXiv:2010.05874*, 2020.

[85] V. Piratla, P. Netrapalli, and S. Sarawagi, "Focus on the common good: Group distributional robustness follows," *arXiv preprint arXiv:2110.02619*, 2021.

[86] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *International conference on machine learning*, PMLR, 2018, pp. 794–803.

[87] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[88] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 264–11 272.

[89] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?" *Advances in neural information processing systems*, vol. 32, 2019.

[90] C. Liang *et al.*, "Super tickets in pre-trained language models: From model compression to improving generalization," *arXiv preprint arXiv:2105.12002*, 2021.

[91] C. Liang *et al.*, "No parameters left behind: Sensitivity guided adaptive learning rate for training large transformer models," in *International Conference on Learning Representations*, 2022.

[92] J. Chen, R. Zhang, J. Guo, Y. Liu, Y. Fan, and X. Cheng, "Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks," *arXiv preprint arXiv:2208.07652*, 2022.

[93] A. Asai *et al.*, “Task-aware retrieval with instructions,” *arXiv preprint arXiv:2211.09260*, 2022.

[94] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, “Signature verification using a” siamese” time delay neural network,” *Advances in neural information processing systems*, vol. 6, 1993.

[95] J. Ni *et al.*, “Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models,” *arXiv preprint arXiv:2108.08877*, 2021.

[96] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR (Poster)*, 2015.

[97] M. Leszczynski, D. Fu, M. Chen, and C. Ré, “Tabi: Type-aware bi-encoders for open-domain entity retrieval,” in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 2147–2166.

[98] T. Nguyen *et al.*, “Ms marco: A human generated machine reading comprehension dataset,” *choice*, vol. 2640, p. 660, 2016.

[99] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, “BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[100] W. Zhang, X. V. Lin, K. Stratos, W.-t. Yih, and M. Chen, “Imprag: Retrieval-augmented generation with implicit queries,” *arXiv e-prints*, arXiv–2506, 2025.

[101] Z. Jiang *et al.*, “Retrieval as attention: End-to-end learning of retrieval and reading within a single transformer,” in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 2336–2349.

[102] J. Zhang *et al.*, “Onegen: Efficient one-pass unified generation and retrieval for llms,” in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 4088–4119.

[103] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev, “Internet-augmented language models through few-shot prompting for open-domain question answering,” *arXiv preprint arXiv:2203.05115*, 2022.

[104] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, “Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions,” in *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[105] Z. Jiang *et al.*, "Active retrieval augmented generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 7969–7992.

[106] M. Chen, X. Chen, and W.-t. Yih, "Few-shot data synthesis for open domain multi-hop question answering," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 190–208.

[107] H. Yang *et al.*, "Memory3: Language modeling with explicit memory," *arXiv preprint arXiv:2407.01178*, 2024.

[108] S. Lu, H. Wang, Y. Rong, Z. Chen, and Y. Tang, "Turborag: Accelerating retrieval-augmented generation with precomputed kv caches for chunked text," *arXiv preprint arXiv:2410.07590*, 2024.

[109] B. Wang *et al.*, "Instructretro: Instruction tuning post retrieval-augmented pretraining," in *International Conference on Machine Learning*, PMLR, 2024, pp. 51 255–51 272.

[110] S. Borgeaud *et al.*, "Improving language models by retrieving from trillions of tokens," in *International conference on machine learning*, PMLR, 2022, pp. 2206–2240.

[111] B. Wang *et al.*, "Shall we pretrain autoregressive language models with retrieval? a comprehensive study," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 7763–7786.

[112] L. Zhang, Y. Yu, K. Wang, and C. Zhang, "Arl2: Aligning retrievers with black-box large language models via self-guided adaptive relevance labeling," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 3708–3719.

[113] A. Asai *et al.*, "Task-aware retrieval with instructions," in *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[114] Y. Lee, M. Kim, and S.-w. Hwang, "Disentangling questions from query generation for task-adaptive retrieval," in *EMNLP (Findings)*, 2024.

[115] H. Oh *et al.*, "Instructir: A benchmark for instruction following of information retrieval models," *arXiv preprint arXiv:2402.14334*, 2024.

[116] O. Weller *et al.*, "Followir: Evaluating and teaching information retrieval models to follow instructions," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025, pp. 11 926–11 942.

[117] W. Wu, Y. Wang, G. Xiao, H. Peng, and Y. Fu, “Retrieval head mechanistically explains long-context factuality,” *arXiv preprint arXiv:2404.15574*, 2024.

[118] Z. Zhao, Y. Ziser, and S. B. Cohen, “Layer by layer: Uncovering where multi-task learning happens in instruction-tuned large language models,” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 15 195–15 214.

[119] N. Muennighoff *et al.*, “Generative representational instruction tuning,” in *ICLR 2024 Workshop: How Far Are We From AGI*, 2024.

[120] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebron, and S. Sanghai, “Gqa: Training generalized multi-query transformer models from multi-head checkpoints,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 4895–4901.

[121] J. Fang *et al.*, “Unimem: Towards a unified view of long-context large language models,” in *First Conference on Language Modeling*, 2024.

[122] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, “Roformer: Enhanced transformer with rotary position embedding,” *Neurocomputing*, vol. 568, p. 127 063, 2024.

[123] W. Zhang, W. Hua, and K. Stratos, “EntQA: Entity linking as question answering,” in *International Conference on Learning Representations*, 2022.

[124] G. Izacard *et al.*, “Unsupervised dense information retrieval with contrastive learning,” *Transactions on Machine Learning Research*, 2022.

[125] T. Kwiatkowski *et al.*, “Natural questions: A benchmark for question answering research,” *Transactions of the Association for Computational Linguistics*, vol. 7, L. Lee, M. Johnson, B. Roark, and A. Nenkova, Eds., pp. 452–466, 2019.

[126] Z. Yang *et al.*, “HotpotQA: A dataset for diverse, explainable multi-hop question answering,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds., Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2369–2380.

[127] M. Chen *et al.*, “Improving in-context few-shot learning via self-supervised training,” in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds., Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 3558–3573.

[128] J. Wei *et al.*, “Measuring short-form factuality in large language models,” *arXiv preprint arXiv:2411.04368*, 2024.

[129] X. Ho, A.-K. Duong Nguyen, S. Sugawara, and A. Aizawa, “Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps,” in *Proceedings of the 28th International Conference on Computational Linguistics*, D. Scott, N. Bel, and C. Zong, Eds., Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6609–6625.

[130] H. Elsahar *et al.*, “T-REx: A large scale alignment of natural language with knowledge base triples,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, N. Calzolari *et al.*, Eds., Miyazaki, Japan: European Language Resources Association (ELRA), May 2018.

[131] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer, “Zero-shot relation extraction via reading comprehension,” in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, R. Levy and L. Specia, Eds., Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 333–342.

[132] J. Hoffart *et al.*, “Robust disambiguation of named entities in text,” in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, R. Barzilay and M. Johnson, Eds., Edinburgh, Scotland, UK.: Association for Computational Linguistics, Jul. 2011, pp. 782–792.

[133] D. A. Ferrucci, “Introduction to “this is watson”,” *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 1–1, 2012.

[134] C. Xiong, J. Callan, and T.-Y. Liu, “Word-entity duet representations for document ranking,” in *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, 2017, pp. 763–772.

[135] F. Hasibi, K. Balog, and S. E. Bratsberg, “Exploiting entity linking in queries for entity retrieval,” in *Proceedings of the 2016 acm international conference on the theory of information retrieval*, 2016, pp. 209–218.

[136] K. Balog, H. Ramampiaro, N. Takhirov, and K. Nørvåg, “Multi-step classification approaches to cumulative citation recommendation,” in *Proceedings of the 10th conference on open research areas in information retrieval*, 2013, pp. 121–128.

[137] R. Reinanda, E. Meij, and M. de Rijke, “Mining, ranking and recommending entity aspects,” in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, pp. 263–272.

[138] Y. Yang, O. İrsoy, and K. S. Rahman, “Collective entity disambiguation with structured gradient tree boosting,” in *Proceedings of the 2018 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 777–786.

[139] B. Slawski, *How google uses named entity disambiguation for entities with the same names*, Accessed: 2021-09-27, Sep. 2015.

[140] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," in *International Conference on Learning Representations*, 2019.

[141] O.-E. Ganea and T. Hofmann, "Deep joint entity disambiguation with local neural attention," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2619–2629.

[142] M. Röder, R. Usbeck, and A.-C. Ngonga Ngomo, "Gerbil–benchmarking named entity recognition and linking consistently," *Semantic Web*, vol. 9, no. 5, pp. 605–625, 2018.

[143] C. Alberti, K. Lee, and M. Collins, "A bert baseline for the natural questions," *arXiv preprint arXiv:1901.08634*, 2019.

[144] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.

[145] L. Derczynski *et al.*, "Analysis of named entity recognition and linking for tweets," *Information Processing & Management*, vol. 51, no. 2, pp. 32–49, 2015.

[146] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum, "Kore: Keyphrase overlap relatedness for entity disambiguation," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 545–554.

[147] M. Röder, R. Usbeck, S. Hellmann, D. Gerber, and A. Both, "N^3-a collection of datasets for named entity recognition and disambiguation in the nlp interchange format.," in *LREC*, 2014, pp. 3529–3533.

[148] A. G. Nuzzolese, A. L. Gentile, V. Presutti, A. Gangemi, D. Garigliotti, and R. Navigli, "Open knowledge extraction challenge," in *Semantic Web Evaluation Challenges*, Springer, 2015, pp. 3–15.

[149] F. Petroni *et al.*, "KILT: A benchmark for knowledge intensive language tasks," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021, pp. 2523–2544.

[150] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 784–789.

[151] A. Akbik, D. Blythe, and R. Vollgraf, “Contextual string embeddings for sequence labeling,” in *Proceedings of the 27th international conference on computational linguistics*, 2018, pp. 1638–1649.

[152] N. Steinmetz and H. Sack, “Semantic multimedia information retrieval based on contextual descriptions,” in *Extended Semantic Web Conference*, Springer, 2013, pp. 382–396.

[153] A. Moro, A. Raganato, and R. Navigli, “Entity linking meets word sense disambiguation: A unified approach,” *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 231–244, 2014.

[154] S. Broscheit, “Investigating entity knowledge in bert with simple neural end-to-end entity linking,” in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019, pp. 677–685.

[155] P. H. Martins, Z. Marinho, and A. F. Martins, “Joint learning of named entity recognition and entity linking,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, 2019, pp. 190–196.

[156] I. Yamada, A. Asai, and H. Hajishirzi, “Efficient passage retrieval with hashing for open-domain question answering,” *arXiv preprint arXiv:2106.00882*, 2021.

[157] L. He, M. Lewis, and L. Zettlemoyer, “Question-answer driven semantic role labeling: Using natural language to annotate natural language,” in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 643–653.

[158] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer, “Zero-shot relation extraction via reading comprehension,” in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 2017, pp. 333–342.

[159] X. Li *et al.*, “Entity-relation extraction as multi-turn question answering,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 1340–1350.

[160] X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li, “A unified mrc framework for named entity recognition,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 5849–5859.

[161] R. Aralikatte, M. Lamm, D. Hardt, and A. Søgaard, “Ellipsis resolution as question answering: An evaluation,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 810–817.

[162] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, "The natural language decathlon: Multitask learning as question answering," *arXiv preprint arXiv:1806.08730*, 2018.

[163] W. Wu, F. Wang, A. Yuan, F. Wu, and J. Li, "Corefqa: Coreference resolution as query-based span prediction," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 6953–6963.

[164] W. Zhang, S. Wiseman, and K. Stratos, "Seq2seq is all you need for coreference resolution," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 11 493–11 504.

[165] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 188–197.

[166] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[167] V. Sanh *et al.*, "Multitask prompted training enables zero-shot task generalization," in *International Conference on Learning Representations*, 2022.

[168] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, "Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes," in *Joint conference on EMNLP and CoNLL-shared task*, 2012, pp. 1–40.

[169] H. Chen, Z. Fan, H. Lu, A. Yuille, and S. Rong, "PreCo: A large-scale dataset in preschool vocabulary for coreference resolution," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 172–181.

[170] S. Toshniwal, P. Xia, S. Wiseman, K. Livescu, and K. Gimpel, "On generalization in coreference resolution," in *Proceedings of the Fourth Workshop on Computational Models of Reference, Anaphora and Coreference*, Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 111–120.

[171] D. Bamman, O. Lewke, and A. Mansoor, "An annotated dataset of coreference in English literature," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Marseille, France: European Language Resources Association, May 2020, pp. 44–54, ISBN: 979-10-95546-34-4.

[172] A. Daza and A. Frank, "A sequence-to-sequence model for semantic role labeling," in *Proceedings of The Third Workshop on Representation Learning for NLP*, 2018, pp. 207–216.

[173] O. Gotoh, “An improved algorithm for matching biological sequences,” *Journal of molecular biology*, vol. 162, no. 3, pp. 705–708, 1982.

[174] H. W. Chung *et al.*, “Scaling instruction-finetuned language models,” *arXiv preprint arXiv: 2210.11416*, 2022.

[175] T. Wolf *et al.*, “Huggingface’s transformers: State-of-the-art natural language processing,” *arXiv preprint arXiv:1910.03771*, 2019.

[176] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, “Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3505–3506.

[177] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, “Zero: Memory optimizations toward training trillion parameter models,” in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, 2020, pp. 1–16.

[178] M. Joshi, O. Levy, L. Zettlemoyer, and D. S. Weld, “Bert for coreference resolution: Baselines and analysis,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 5803–5808.

[179] J. Yu, A. Uma, and M. Poesio, “A cluster ranking model for full anaphora resolution,” in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 2020, pp. 11–20.

[180] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, “Spanbert: Improving pre-training by representing and predicting spans,” *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020.

[181] P. Xia, J. Sedoc, and B. Van Durme, “Incremental neural coreference resolution in constant memory,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 8617–8624.

[182] S. Toshniwal, S. Wiseman, A. Ettinger, K. Livescu, and K. Gimpel, “Learning to ignore: Long document coreference with bounded memory neural networks,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 8519–8526.

[183] V. Dobrovolskii, “Word-level coreference resolution,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 7670–7675.

[184] P. Xia and B. Van Durme, "Moving on from ontonotes: Coreference resolution model transfer," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 5241–5256.

[185] R. Thirukovalluru, N. Monath, K. Shridhar, M. Zaheer, M. Sachan, and A. McCallum, "Scaling within document coreference to long texts," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 3921–3931.

[186] Z. Wu and M. Gardner, "Understanding mention detector-linker interaction in neural coreference resolution," in *Proceedings of the Fourth Workshop on Computational Models of Reference, Anaphora and Coreference*, 2021, pp. 150–157.

[187] B. Bohnet *et al.*, "Attributed question answering: Evaluation and modeling for attributed large language models," *arXiv preprint arXiv:2212.08037*, 2022.

[188] T. Gao, H. Yen, J. Yu, and D. Chen, "Enabling large language models to generate text with citations," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2023.

[189] Z. Liu, K. Zhang, C. Xiong, Z. Liu, and M. Sun, "Openmatch: An open source library for neu-ir research," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2531–2535.

[190] H. Yen, T. Gao, and D. Chen, "Long-context language modeling with parallel context encoding," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 2588–2610.

[191] Z. Zhang *et al.*, "H2o: Heavy-hitter oracle for efficient generative inference of large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 34 661–34 710, 2023.

[192] A. Köpf *et al.*, "Openassistant conversations - democratizing large language model alignment," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[193] S. Reddy, D. Chen, and C. D. Manning, "CoQA: A conversational question answering challenge," *Transactions of the Association for Computational Linguistics*, vol. 7, L. Lee, M. Johnson, B. Roark, and A. Nenkova, Eds., pp. 249–266, 2019.

[194] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner, "DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs," 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)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 2368–2378.

[195] A. Trischler *et al.*, “NewsQA: A machine comprehension dataset,” in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, P. Blunsom *et al.*, Eds., Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 191–200.

[196] Q. Jin, B. Dhingra, Z. Liu, W. Cohen, and X. Lu, “PubMedQA: A dataset for biomedical research question answering,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2567–2577.

[197] A. Rogers, O. Kovaleva, M. Downey, and A. Rumshisky, “Getting closer to ai complete question answering: A set of prerequisite real tasks,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 8722–8731, Apr. 2020.

[198] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for SQuAD,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, I. Gurevych and Y. Miyao, Eds., Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 784–789.

[199] D. Chen, J. Bolton, and C. D. Manning, “A thorough examination of the CNN/Daily Mail reading comprehension task,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Erk and N. A. Smith, Eds., Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 2358–2367.