RANKING AND RETRIEVAL UNDER SEMANTIC RELEVANCE

by

Tongfei Chen

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland

October 2020

Abstract

This thesis presents a series of conceptual and empirical developments on the *ranking* and *retrieval* of candidates under semantic relevance. Part I of the thesis introduces the concept of uncertainty in various semantic tasks (such as recognizing textual entailment) in natural language processing, and the machine learning techniques commonly employed to model these semantic phenomena. A unified view of ranking and retrieval will be presented, and the trade-off between model expressiveness, performance, and scalability in model design will be discussed.

Part II of the thesis focuses on applying these ranking and retrieval techniques to text: Chapter 3 examines the feasibility of ranking hypotheses given a premise with respect to a human's subjective probability of the hypothesis happening, effectively extending the traditional categorical task of natural language inference. Chapter 4 focuses on detecting situation frames for documents using ranking methods. Then we extend the ranking notion to retrieval, and develop both sparse (Chapter 5) and dense (Chapter 6) vector-based methods to facilitate scalable retrieval for potential answer paragraphs in question answering.

Part III turns the focus to mentions and entities in text, while continuing the theme on ranking and retrieval: Chapter 7 discusses the ranking of fine-grained types that an entity mention could belong to, leading to state-of-the-art performance on hierarchical multi-label fine-grained entity typing. Chapter 8 extends the semantic relation of coreference to a cross-document setting, enabling models to retrieve from a large corpus, instead of in a single document, when resolving coreferent entity mentions.

Primary Reader and Advisor: Prof. Benjamin Van Durme

Secondary Reader: Prof. David Yarowsky, Prof. João Sedoc

Acknowledgments

First and foremost, I'm immensely grateful to my advisor Prof. Benjamin Van Durme. Ben mentored me for six years, offering unwavering support through my roller coaster ride as a Ph.D. student, and taught me to conduct meaningful original research, to present ideas and write papers clearly, and to always look at the bigger picture from wider perspectives. Ben also offered invaluable advices to me, not limited to academic life. I would also like to thank Prof. David Yarowsky and Prof. João Sedoc for serving as members of my thesis committee. Their advice had been immensely helpful for the completion of this thesis.

I have been very fortunate to collaborate with my colleagues and co-authors at CLSP. From chitchat in the hallways of Hackerman Hall to technical discussion in front of white-boards, I'm thankful to all of you: Yunmo Chen, Ryan Culkin, Shuoyang Ding, Seth Ebner, Zhengping Jiang, Nils Holzenberger, Huda Khayrallah, Zhongyang Li, Arya McCarthy, Nanyun Peng, Adam Poliak, Pushpendre Rastogi, Rachel Rudinger, Keisuke Sakaguchi, Rashmi Sankepally, Elias Stengel-Eskin, Patrick Xia, Hainan Xu, Yiming Wang.

Life at Hopkins was much enjoyable because of the brilliant cohort of graduate students here, with whom I have had not only intellectually stimulating conversations, but also lasting friendships. Thank you to Anton Belyy, Yunmo Chen, Shuoyang Ding, Seth Ebner, Matthew Francis-Landau, Jonathan Jones, Chu-Cheng Lin, Huda Khayrallah, Arya McCarthy, Sabrina Mielke, Nanyun Peng, Guanghui Qin, Pushpendre Rastogi, Adithya Renduchintala, Rachel Rudinger, Keisuke Sakaguchi, Tim Vieira, Hainan Xu, Dingquan Wang, Yiming Wang, Travis Wolfe, Fei Wu, Mo Yu, Sheng Zhang, and many others.

A special thanks to Shuoyang Ding, for sharing an apartment for six years, and for introducing me to the excellent workflow management tool `ducttape` by Jonathan Clark (thanks Jonathan!), that made my convoluted experiment pipelines so much easier to manage, and without which I would definitely graduate much later.

Off-campus, I am lucky to have the opportunity to work with Dr. Jiří Navrátil at IBM Research on various aspects of confidence estimation, and Dr. Lambert Mathias at Amazon on dialogue understanding. Thank you for giving me such great internship opportunities.

I'd like to thank Center for Language and Speech Processing and the Department of Computer Science for providing institutional support. Thanks to Ruth Scally, Jennifer Linton, Zack Burwell, and Kim Franklin for providing help on many occasions involving administrative matters. Additionally, I'd like to thank the Human Language Technology Center of Excellence for providing computational resources that facilitates my research.

Finally, my deep and sincere gratitude to my family for their steadfast and unparalleled love and support. This journey would not have been possible if not for them.

Contents

List of Tables

List of Figures

Part I

Fundamentals

Chapter 1

Introduction

Various tasks in the general field of artificial intelligence (AI) can be formulated as a ranking problem on a set of elements. These elements may include labels (e.g. types in an ontology), sentences, paragraphs, documents, mentions, or entities in knowledge bases (KBs), etc.

Some examples of ranking problems that arise in NLP include:

- Document search: given a search query, returns documents that are most relevant with respect to the query (i.e. Google Search);
- Textual question answering: given a natural language question, return sentences that contain the answer to the query;
- Natural language inference or plausiblity: given a context, determine which outcome is most probable given the context.

Take a well-studied task, corpus-based question answering (QA), as an example. Given a natural language question, a system finds sentences in a corpus that answers a given question. Current research in corpus-based QA is typically modeled as a two-step process: (**1**) *triaging*: finding a candidate set of passages (e.g., documents or sentences in a corpus) that may contain an answer to a question, followed by (**2**) *downstream models*: a downstream candidate reranking / selection model that refines and extracts the answer from the initial set.

The reason for employing this two-step process, instead of a direct run of the downstream model on the whole corpus, is the problem of *scalability*: A typical QA system like Watson (Ferrucci et al., 2010) processes millions of documents to answer questions, yet the response time should be restricted to seconds, or even below a second. Directly running the downstream model (which, in recent years, has become quite sophisticated) on the whole text corpus could be prohibitively infeasible – the runtime complexity is linear to the number of candidates. This means that knowledge in the text corpus has to be *indexed*, i.e. preprocessed and organized offline in some fashion, to enable fast searching and access.

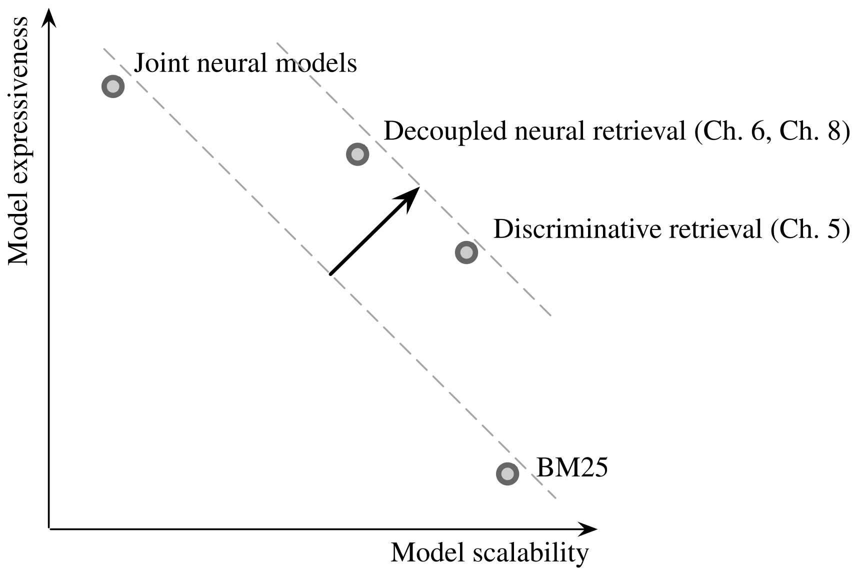
In this thesis, we develop approaches and methods that addresses both of these two problem, triaging candidates for *retrieval*, and *ranking* the candidates for various problems that arise in semantics. We seek to address the following problems under a unified perspective of ranking and retrieval under some semantic relevance:

- How can we train a system that learns to rank under various semantic relevance?

- How can a system learn representations of objects (i.e. sentences, mentions, etc.) so that such representations facilitate scalable, efficient retrieval in very large corpora?

Ranking We explore modeling choices that explicitly account for uncertainties in various semantic problems, modeling them as *ranking* problems. Specifically, in Chapter 3, we extend the traditional categorical classification task of natural language inference to a scalar prediction task, namely *uncertain natural language inference* (UNLI), where the model is asked to predict the human subjective probability of the hypothesis happening given a premise. We explore various modeling techniques, such as regression and ranking to the problem, illustrating modeling abilty to predict subjective probabilities that correlates well with humans. In Chapter 4, we turn to examine the detection of *situation frames* in text, where models should respond to the emergence of natural disasters or social crises mentioned in documents, under low-resource language scenarios. We demonstrate various ranking methods to ameliorate biases generated from data annotation from crowdsourcing workers. In Chapter 7, we examine the semantic relation of a mention being an instance of an entity type, by designing a ranking method over fine-grained entity types in a tree-structured type ontology, leading to state-of-the-art results across multiple datasets.

Retrieval Current systems that perform well on ranking (e.g. state-of-the-art question answering systems (D. Chen et al., 2017), state-of-the-art coreference resolution systems (Joshi, D. Chen, et al., 2020)) does not scale to large corpora: they are only able to operate



Model expressiveness
Joint neural models
Decoupled neural retrieval (Ch. 6, Ch. 8)
Discriminative retrieval (Ch. 5)
BM25
Model scalability

Figure 1.1: Trade-off between model expressiveness and scalability.

in a small candidate set that contains about hundreds of paragraphs under limited time constraints (e.g. less than a second). Conversely, retrieval systems that can scalably retrieve from millions or billions of documents usually resort to shallow, lexical features to ensure fast performance.

There exists a tradeoff between model expressiveness and model scalability: a more complex model lends to more expressiveness of modeling power, resulting in more performant models; but such complex modeling requires heavyweight computation that is simply not feasible for scalability concerns. This could be illustrated in Figure 1.1, where one goal of this thesis is to explore methods to advance the Pareto frontier.

To address these problems, in Chapter 5, we extend the traditional bag-of-words retrieval model to a *bag-of-features* one for the task of triaging in question answering, with linguistic features customizable and feature weights learned, from a *learning-to-rank* model based on question answering datasets, leading to improved recall in triaging systems. In Chapter 6, we explore pure neural solutions for the triaging problem that uses approximate maximum inner product search methods for dense representations. Turning our focus to mentions again, in Chapter 8, we extend the span-ranking model in state-of-the-art coreference resolution systems to a cross-document retrieval setting, enabling coreference resolution systems to attend to mentions across documents.

The various chapters of this thesis can be summarized in Figure 1.2.

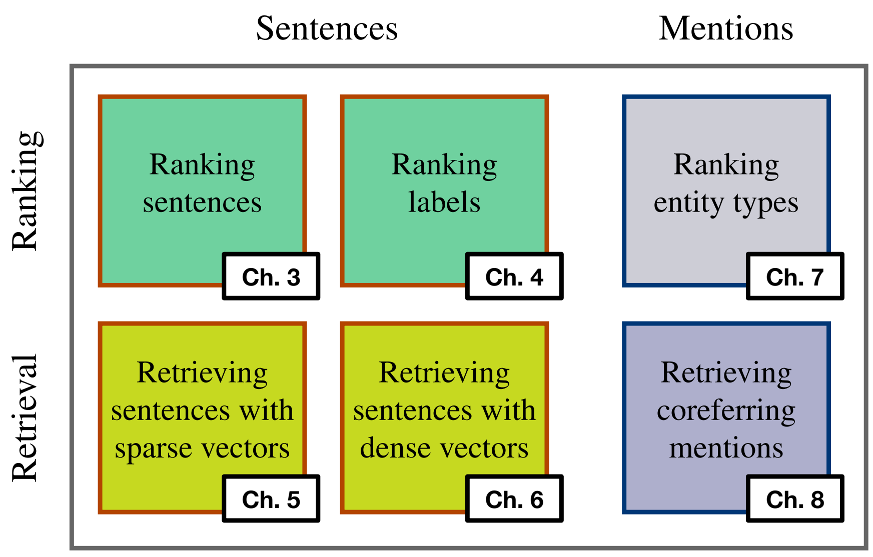


Figure 1.2: Overall structure of this thesis.

Chapter 2

Background

2.1 Ranking

Many tasks in information retrieval (IR) and natural language processing (NLP) have *ranking* as a central problem. These include the canonical IR task document retrieval, and other NLP tasks such as entity search or question answering, and extends to other AI tasks such as online advertisement, dialogue generation, or content-based image retrieval (H. Li, 2014).

The problem of ranking can be formulated as the following mathematical problem: Given a query $q \in \mathcal{Q}$, a set of candidates $C = \{c_1, \cdots, c_N\} \subseteq \mathcal{C}$,[1] rank the elements in C based on the information from the query q and the set of candidates C.

This is most often modeled by a **relevance function**, or *scoring* function $F : \mathcal{Q} \times \mathcal{C} \rightarrow$

[1] A note on the notation here: C is the candidate set, and $\mathcal{C}$ is the set of all possible candidates. For example, C could be all sentences in Wikipedia, where $\mathcal{C}$ would then be the set of all possible sentences.

$\mathbb{R}$: we say that the relevance, or the score between query q and a specific candidate c is $F(q, c)$. The set of candidates C is thus ranked, and can be sorted, under the function $F(q, \cdot)$.

2.1.1 Document Retrieval as an Example

Let's start our discussion with the traditional document retrieval task as an example.

One of the oldest ranking model for document retrieval is TF-IDF, a numerical statistic that models how important a term t is to a document d in a corpus D. It is a product of two factors: **term frequency** (TF), which is proportional to the number of times a term occurs in a document (Luhn, 1957); and **inverse document frequency** (IDF), also known as the *specificity of a term*, can be quantified as an inverse function of the number of documents in which it occurs (Jones, 1972).

TF-IDF is used as the relevance function to rank candidate passages given a query. It has many incarnations with variations in computation details, but a widely used instance is the Okapi BM25 (Robertson, Walker, et al., 1994) due to their robustness in *ad hoc* retrieval:

$$F(q, d) = \sum_i \mathrm{idf}(q_i) \cdot \frac{(k_1 + 1) \cdot \mathrm{freq}(q_i, d)}{\mathrm{freq}(q_i, d) + k_1 \left(1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}}\right)} \tag{2.1}$$

where q is a query, q_i is the i-th token in q, d is any candidate document, and $|d|$ is its length, avgdl is the average document length, $\mathrm{idf}(q_i)$ is the IDF (inverse document frequency) of token q_i, and freq(q_i, d) is the frequency (number of occurrences) of token q_i in document d. Values $k_1 = 1.2$ and $b = 0.75$ are hyperparameters.

In another thread of work, the ranking model is defined as a conditional probability distribution $F(q, d) = P(q \mid d)$ under a language model (Ponte and Croft, 1998; Zhai and Lafferty, 2001). Both TF-IDF based methods and language model based methods requires no training (only tuning a few parameters is necessary, for example, the k_1 and b in BM25 Equation 2.1).

These ranking methods work well for document retrieval since the relevance function models word overlap. However, there is a trend that arises in the IR community to employ machine learning techniques such as *learning-to-rank*, to automatically construct such a relevance function $F(q, c)$. This is useful because:

- **Model expressiveness:** We are motivated to model more complex, semantic relevance functions that goes beyond what can be captured by mere word overlap;
- **Feasibility of obtaining training data:** We can obtain training data for ranking, for example, clickthrough data aggregated by web search engines.

These issues motivate the use of supervised learning methods to train a relevance function, which is what we are going to discuss below.

2.1.2 Supervised Learning to Rank

For a supervised learning setup, a training dataset is required. In the field of learning-to-rank, each training sample is set to be (x, y) where $x = (q, c)$. Here $q \in \mathcal{Q}$ is the query, $c \in \mathcal{C}$ is the candidate object, and $y \in \mathcal{Y}$, where the labels represent grades about how relevant q is to c. Usually $\mathcal{Y}$ is set to be a discrete set $\{0, 1, \cdots, L-1\}$, where 0 stands for "irrelevant", 1 stands for "somewhat relevant", $\cdots$, and $L-1$ stands for "very relevant".

The label set $\mathcal{Y}$ has a total ordering endowed, which is usually denoted as "$\succ$": $i \succ j$ is read as "i is a higher relevance grade than j." We will extend this graded, Likert scale-styled label set to a scalar value in $\mathbb{R}$ in Chapter 3.

There are mainly three classes of ranking methods, namely *pointwise*, *pairwise*, and *listwise* ranking. The main difference lies in the loss function employed:

- Pointwise ranking: The loss takes each sample independently, reducing the ranking problem to a traditional classification or regression problem.

$$L(\mathcal{D}) = \sum_{(q,c,y)\in\mathcal{D}} \ell(q, c, y); \tag{2.2}$$

- Pairwise ranking: The loss takes a pair of samples at a time, and learns to compare these two to decide which one is of higher relevance.

$$L(\mathcal{D}) = \sum_{y^+ \succ y^-} \ell(q^+, c^+, y^+, q^-, c^-, y^-); \tag{2.3}$$

- Listwise ranking: The loss takes a query and all of its associated candidates and their labels, and jointly considers the rank of all the candidates in the list.

$$L(\mathcal{D}) = \sum_q \ell(q, (c_1, \cdots, c_n), (y_1, \cdots, y_n)) \tag{2.4}$$

It is observed that pairwise and listwise approach usually outperform the pointwise counterpart (H. Li, 2014). Pointwise and pairwise ranking can easily be incorporated in a neural model (as compared to listwise methods) that can be trained end-to-end, hence will be extensively used and discussed in this thesis. One can refer to Duh (2009) for an extensive discussion on ranking methods.

2.1.3 Pointwise Ranking

In the pointwise ranking approach, the ranking problem is reduced to standard classification or regression. Therefore, the structure of the ranked list is ignored in this approach.

More specifically, the pointwise approach predicts the score of an instance (q, c) using the relevance function $F(q, c)$. Depending on the type of the label y,

- if it is a binary label $\{0, 1\}$ designating irrelevance/relevance, the ranking problem is effectively reduced to a binary classification problem;

- if it is a set of labels $\{0, 1, \cdots, L - 1\}$ representing graded relevance, the ranking problem turns into an *ordinal classification* problem;

- or if the label is a scalar variable (e.g. in $[0, 1]$ or $\mathbb{R}$), the ranking problem is effectively a regression problem.

Once the model is reduced to a binary classification or regression problem, standard machine learning models such as logistic regression and SVMs can be employed.

2.1.4 Pairwise Ranking

Contrary to the pointwise approach, instead of taking one sample for consideration at a time, the pairwise ranking takes two. Instead of directly answering the question "what is the label for this sample", we ask "of these two samples, which one is more relevant." It hence reduces the problem of ranking to a classification problem on pairs.

A central concern in pairwise ranking is the construction of a *preference pair set*. A preference pair set contains sample pairs where one is more relevant than the other. These preference pairs is therefore used as the training set for the pairwise classification problem that decides which one is "better", or more relevant. ee When constructing the preference pair set, one often employs *negative sampling*: For each relevant (or positive) query/candidate pair, a set of negative samples are sampled to serve as contrastive samples for the model to learn. Usually different sampling techniques are used for different problems, and tuned to the need of the specific task, for example, the number of negative samples is usually a hyperaparameter tuned using a dev set.

Given the pairwise preference set $\mathcal{R} = \{q, c_+, c_-\}$ where $c_+ \succ c_-$ (c_+ ranks higher than c_- under query q), the model can be trained using various losses. Under a cross entropy loss, the model is essentially RankNet (Burges et al., 2005), where the probability of c_+ ranking higher than c_- is maximized:

$$P(c_+ \succ c_-) = \frac{\exp(F(q, c_+) - F(q, c_-))}{1 + \exp(F(q, c_+) - F(q, c_-))} \ . \tag{2.5}$$

Or, such a model can be learned under a primal SVM formulation. This is equivalent to the RankSVM method (Herbrich, Graepel, and Obermayer, 1999):

$$L = \sum_{(q, c_+, c_-) \in \mathcal{R}} \max\{0, \xi - F(q, c_+) + F(q, c_-)\} \ , \tag{2.6}$$

where ξ is a margin hyperparameter that denotes the desired margin between the relevance score of positive and negative candidates.

2.2 Retrieval

The task of retrieval is closely tied to that of ranking. In the retrieval scenario, we are not concerned with ranking the whole candidate set C as what we consider under the ranking scenario. Instead, we are only concerned with the most relevant k elements: we wish to retrieve the top-k elements in C that score the highest under the relevance function F given a query q. Abstractly, we are solving the following problem:

$$C'_q = \underset{c \in C}{\arg \operatorname{top}_k} \ F(q, c) \subseteq C \ , \tag{2.7}$$

The retrieved top-k set $C'_q \subseteq C$. Here q is the query, $c \in C$ is a candidate, C is the set of candidates to retrieve from, k is the number of top items to retrieve and $F(\cdot, \cdot)$ is the relevance function. We denote "$\arg\text{top}_k$" as a generalization to "$\arg\max$": selecting top-k instead of just selecting 1.

2.2.1 Triaging as Approximate Retrieval

As we have discussed in Chapter 1, question answering systems are usually modeled as a two-step process consisting of *triaging* (retrieval) and *reranking* steps, owing to scalability concerns. This two-step process can be generalized to various information extraction (IE) problems, whenever the candidate set is too large (millions, or even billions) to run a model over exhaustively, could use a triaging model to scale it. Lots of tasks in AI can be cast as a ranking problem under the aforementioned triaging framework:

- **Documents as candidates:** The most traditional task of information retrieval (IR), where the query is a set of search terms and the candidate set is a set of documents, is a classical example in this case. The system selects candidate documents that have word overlap with the given search terms.
- **Sentences as candidates:** Corpus-based QA is an example as we have elaborated before. Additional examples include finding in a large corpus semantically related sentences (answering a question; being a paraphrase to the query; entailing the query; or being entailed by the query etc.) given a sentence.

- **Knowledge base entities as candidates:** An example of this is the task of knowledge base question answering (KBQA), in which the system retrieves entities in a knowledge graph given a natural language question. We'll use an example in Yao and Van Durme (2014) here: given the question "*Who is Justin Bieber's sister?*", the system should return the entity "*Jazmyn Bieber*" in the knowledge graph. Another example is entity linking (EL), where the system links a mention in the text (e.g. the text snippet "*Justin Bieber*" in a snippet) to an entity in a KB that it refers to.

- **Mentions in text as candidates:** We define coreferent mention retrieval (CMR) (Sankepally et al., 2018) as an information retrieval task in which one passage mentioning a specifc entity is presented as an example, and the system's task is to find all other sentences in the test collection in which that same entity is mentioned.

- **Tasks outside of NLP:** There's been a series of work outside of NLP that are highly related to this triaging problem. In speech processing, the task of speech term discovery (Park and Glass, 2008; Jansen, Church, and Hermansky, 2010; Jansen and Van Durme, 2011) aims to find repeated similar acoustic patterns in a collection of speech in an unknown language. Also in computer vision research, there is the task of content-based image retrieval (CBIR) (Smeulders et al., 2000; Lew et al., 2006; Wan et al., 2014), where the system retrieves from a large image set similar images to a query image (an example is Google Image Search).

These tasks could be summarized in the table below, which could gives us a clearer idea of the ubiquity of the triaging task in AI research.

Task	**Query** $q \in \mathcal{Q}$	**Candidate set** $C \subseteq \mathcal{C}$
Information retrieval	Search terms	Documents
Text-based question answering	Question	Sentences
Finding paraphrases	Sentence	Sentences
Textual entailment	Sentence	Sentences
Textual abduction	Sentence	Sentences
Knowledge base question answering	Question	Entities
Entity linking	Mention	Entities
Coreferent mention retrieval	Mention	Mentions
Retrieval-based dialog generation	Context	Sentence
Speech term discovery	Speech snippet	Speech snippets
Text-based image retrieval	Search terms	Images
Content-based image retrieval	Image	Images

Table 2.1: Problems that could use a triaging system to scale.

Under scrutiny, we can formulate the triage-then-rerank setup as an *approximation* of exact retrieval. Instead of exact ranking using the relevance function Equation 2.7:

$$R_F(q, C) = \underset{c \in C}{\arg\operatorname{top}_k}\, F(q, c) \tag{2.8}$$

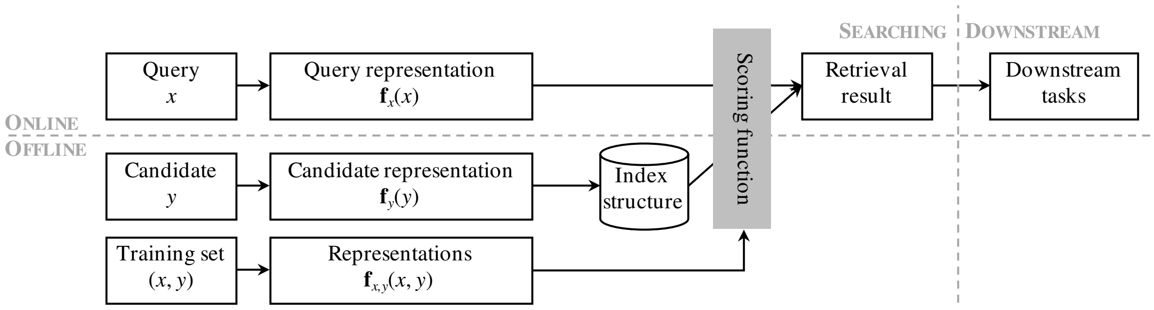


Figure 2.1: General scheme of retrieval problems.

we approximate this with the triage-then-rerank process:

- **Triaging:** We first filter the very large candidate set C with a *surrogate* relevance function $\tilde{F}$ that approximate the better relevance function F, so that computation is feasible for downstream models. This results in a reduced candidate set with a much smaller size: $|\tilde{C}| \ll |C|$:

$$\tilde{C} = \underset{c \in C}{\arg\operatorname{top}_{\tilde{k}}} \tilde{F}(q, c) \tag{2.9}$$

  This should be very efficient (ideally *sublinear* with respect to $|C|$) to run for this step to be computationally feasible.

- **Reranking:** Then we rerank the triaged set $\tilde{C}$ with a better relevance function F, and then take the top k:

$$\hat{C} = \underset{c \in \tilde{C}}{\arg\operatorname{top}_{k}} F(q, c) \tag{2.10}$$

  We can afford to use more heavyweight and complex models since it is feasible to traverse the reduced set, with a linear complexity of $\mathcal{O}(|\tilde{C}|)$.

As we discussed before, for many of these retrieval problems under *semantic relevance*, traditional retrieval methods like BM25 lacks model expressivenss. We turn to learning-to-rank methods to automatically construct such relevance functions from training sets.

A schematic illustration of a learned triage-then-rerank system can be found in Figure 2.1.

2.2.2 Efficient Triaging

How can we make the triage step efficient? Under scrutiny, the approximate scoring function $\tilde{F}$ takes the following form:

$$\tilde{F}(q, c) = S(\mathbf{f}_Q(q), \mathbf{f}_C(c)) \ . \tag{2.11}$$

Such a function can be illustrated in the form of computation graphs in Figure 2.2. The computation of a relevance function can be decomposed into two stages: **decoupled computation**, the independent computation of the representations of the queries and candidates, resulting in the representations of a query $\mathbf{f}_Q(q) \in \mathcal{R}_Q$, and the representations of candidates $\mathbf{f}_C(c) \in \mathcal{R}_C$ for all $c \in C$;[2] and **coupled computation**, where the final relevance score based on the two representations is computed as in Equation 2.11. Therefore such a triaging system could be decomposed into the following three subproblems:

- **Query representation ($\mathbf{f}_Q : \mathcal{Q} \to \mathcal{R}_Q$):** The query q has to be represented in some form $\mathbf{f}_Q(q)$ to enable computation. It could be a bag-of-words in the IR task (essen-

[2] Here $\mathcal{R}_Q$ and $\mathcal{R}_C$ are the space of the representations of queries and candidates respectively. For example, under a dense vector-based representation, $\mathcal{R}_Q = \mathbb{R}^d$.

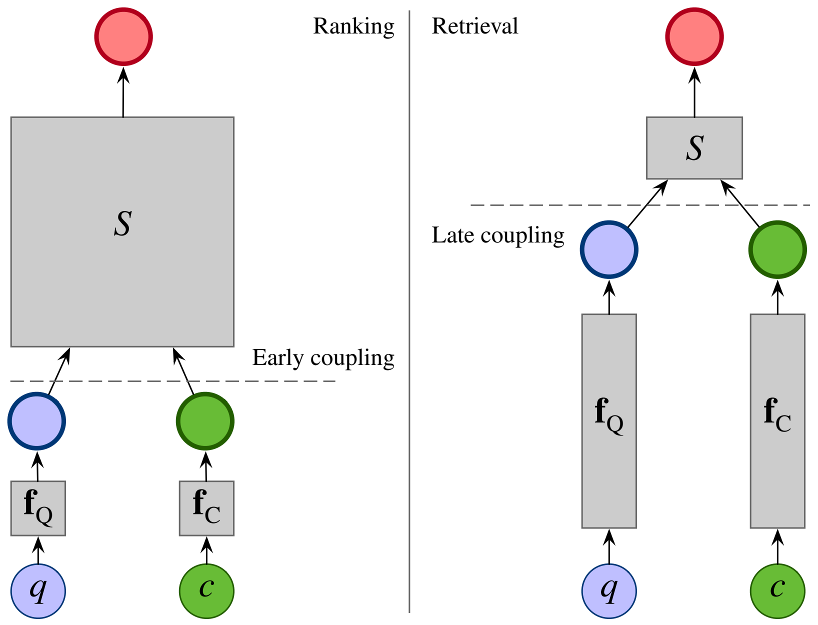


Figure 2.2: The tradeoff between earlier coupling and later coupling for retrieval and reranking models. Here gray boxes represent functions, and the sizes of which are qualitative representations of the computation resources required to compute that function (e.g. number of parameters, etc.). The larger the boxes are, the more complicated those functions are.

tially a sparse vector representation of the query); or in the case of image retrieval, the image is precomputed to yield a *code* (could be a dense vector representation, or a bit sequence).

- **Candidate representation ($\mathbf{f}_C : \mathcal{C} \rightarrow \mathcal{R}_C$):** Similarly, candidates should also be preprocessed and indexed *offline*: each candidate $c \in C$ will be used to generate some representation $\mathbf{f}_C(c)$.

- **Final scoring function ($S : \mathcal{R}_Q \times \mathcal{R}_C \rightarrow \mathbb{R}$), search algorithms and data structures:** There should be a data structure that stores a large number of candidate representations $\{\mathbf{f}_c(c) \mid c \in C\}$ that enables a fast search algorithm that computes Equation 2.11 under the similarity function $S(\cdot, \cdot)$.

The earlier the coupling is, the more complicated the final scoring function S would be, hence leading to better model expressiveness and performance, but unavoidably reducing computational efficiency, making models less scalable. Conversely, the later the coupling is, the final scoring function S will be simple (e.g. cosine distance or inner product), moving large amounts of representation computation offline to be indexed. A simple final scoring function S allows for the design of very efficient searching algorithms (see sections below). This makes models easier to scale, but hurts the performance.

2.2.3 Retrieval over Sparse Vectors

We'll investigate a widely used relevance function, Okapi BM25 (Robertson, Walker, et al., 1994) under the framework we just described.

Following the formulation of linear feature-based IR models (Metzler and Croft, 2007), the relevance function in Equation 2.1 can be reformulated as an inner product between a feature vector $\mathbf{f}_q(q)$ on query q and another feature vector $\mathbf{f}_d(d)$ on candidate document d. These feature vectors are sparse and easily indexed. Under the notation of sparse vectors

Queries	$\mathcal{Q}$	Search terms
	$\downarrow$	Bag-of-words
Query representations	$\mathcal{R}_Q$	$\mathbb{R}^V$: Sparse vectors
Candidates	$\mathcal{C}$	Documents
	$\downarrow$	Bag-of-words
Candidate representations	$\mathcal{R}_C$	$\mathbb{R}^V$: Sparse vectors
Scoring function	$S(\cdot,\cdot)$	Sparse inner product
Indexing structure		Postings lists

Table 2.2: The three subproblems identified in a sparse-vector based retrieval system.

as associative arrays, the BM25 scoring function can be expressed in the following form:

$$\mathbf{f}_q(q) = \{t : 1\}_{t \in q} \; ; \tag{2.12}$$

$$\mathbf{f}_d(d) = \left\{ t : \mathrm{idf}(t) \cdot \frac{\mathrm{freq}(t,d) \cdot (k_1 + 1)}{\mathrm{freq}(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{\mathrm{len}(d)}{\mathrm{avgdl}}\right)} \right\}_{t \in d} \; ; \tag{2.13}$$

$$F(q,d) = \mathbf{f}_q(q) \cdot \mathbf{f}_d(d) \; . \tag{2.14}$$

We can see that under Eqs. (2.12, 2.13), Eq. (2.14) is equivalent to Eq. (2.1). By this featurization we cast the BM25 scoring function as a linear ranking function.

Retrieval under an inner product function in Eq. (2.14) can be implemented efficiently (sublinear time complexity – not all candidates are going to be traversed) using the classical data structure known as *postings lists*.

There has been some work to generalize this to include richer features: we will discuss these in Chapter 5.

2.2.4 Retrieval over Dense Vectors

The retrieval problem when applied to dense vectors, becomes an approximate nearest neighbor search (ANNS) or maximum inner product search (MIPS) problem.

Normally triaging with dense vectors are reduced to a nearest neighbor search (NNS) problem, where we define the scoring function to be a distance

$$F(q, c) = S(\mathbf{f}_q(q), \mathbf{f}_c(c)) \ , \tag{2.15}$$

where $S(\cdot, \cdot)$ is a distance measure (usually L_1 or L_2 distance, which we seek to minimize) or a similarity measure (cosine similarity, inner product, etc., which we seek to maximize).

When the dimensionality of $\mathbf{f}_q(q)$ and $\mathbf{f}_c(c)$ is low, the problem is easily solved sublierly ($O(\log|Y|)$) using data structures such as k-d trees (Bentley, 1975) or R-trees (Roussopoulos, Kelley, and Vincent, 1995). However, when the dimensionality of the vector grows (usually more than 100 in typical neural embeddings), we encounter the "*curse of dimensionality*" (Bellman, Corporation, and Collection, 1957) which greatly complicates the search. Usually we resort to approximate nearest neighbor search (as a trade-off for scalability and speed) instead of an optimal search algorithm in high-dimensional settings.

One possible way of dealing with high-dimensional nearest neighbor search is a family of closely related algorithms called locality sensitive hashing (LSH) (Gionis, Indyk, and Motwani, 1999). In some versions of LSH, every vector $\mathbf{x} \in \mathbb{R}^d$ is hashed to a vector $\mathbf{h}(\mathbf{f}_q(x)) \in \mathbb{B}^k$, where $\mathbb{B}$ could be $\{0, 1\}$ (being the Hamming space, where very fast retrieval can be performed since modern computer architectures can compute Hamming distances in just a few CPU instructions), and there is a distance in $\mathbb{B}^k$ space $d_H(\cdot,\cdot)$ such that $d_H(\mathbf{h}(\mathbf{f}_q(x)),\mathbf{h}(\mathbf{f}_c(y))) \approx d(\mathbf{f}_q(x),\mathbf{f}_c(y))$ when the distance is small, and solving the problem in H^k space is usually much easier and offers a sublinear $O(\log|Y|)$ algorithm.

Various LSH algorithms have been proposed for different distance functions. These include L_2 (Euclidean) distance (Andoni and Indyk, 2006), cosine similarity (Charikar, 2002) and more recently, inner product (Neyshabur and Srebro, 2015; Shrivastava and P. Li, 2014), among others.

Another way of dealing with NNS with dense vectors is using space partitioning methods (that partitions the space into a hierarchical tree of subspaces) (Dasgupta and Sinha, 2013). These methods include randomized k-d trees (Silpa-Anan and Hartley, 2008), in which space is split at each level by a coordinate chosen randomly among those whose data set in that subspace shows the greatest variances. Multiple trees are constructed to minimize searching approximation errors. Another method is the hierarchical k-means tree (Nistér and Stewénius, 2006), which recursively cluster the dataset by k-means algorithm into a tree where nodes in a common subtree are close in the space.

Space partition methods (constructing trees) suffer from a performance issue: at search time, from the root node to the actual leaf, at every node encountered a similarity function evalation has to be invoked for that node and the given query. Fast search on Hamming distances are desired, leading to a new family of techniques called product quantization (PQ) (Jégou, Douze, and Schmid, 2011), where the original high-dimensional space is

Queries	$\mathcal{Q}$	Question
	$\downarrow$	Neural encoder
Query representations	$\mathcal{R}_Q$	$\mathbb{R}^d$: Dense vectors
Candidates	$\mathcal{C}$	Passages
	$\downarrow$	Neural encoder
Candidate representations	$\mathcal{R}_C$	$\mathbb{R}^d$: Dense vectors
Scoring function	$S(\cdot,\cdot)$	Dense inner product
Indexing structure		Product quantization

Table 2.3: The three subproblems identified in a dense-vector based retrieval system.

decomposed into Cartesian products of subspaces, with each subspace quantized by k-means. This achieves good search performance and efficient runtime. An improvement is the optimized product quantization (OPQ) (Ge et al., 2013), where the space decomposition is trained by a subset of the original dataset so that the space decomposition is less lossy. Scalable NNS systems that supports millions to billions of candidates has been built, with the notable example of FAISS (Johnson, Douze, and Jégou, 2017), which is used in this thesis' research.

2.2.5 Evaluation Metrics

To evaluate the performance of a ranking model, one usually takes the ground truth list of candidates and compare it against the predicted list of candidates.

Commonly used metrics include mean average precision (MAP), mean reciprocal rank (MRR), or precision and recall at k (P@k, and R@k). These metrics are computed by averaging the score for each query in a test set: for example, the MAP score is the average of the average precision (AP) scores for each query in the test set.

These common metrics will be briefly described here, where the formula focuses on a single query instead of the test set, since the score of the test set is the average of all the scores of each query.

- **Precision at *k*** (P@k): This is the proportion of the top-k candidate retrieved that are actually relevant to the query. This is especially relevant to modern web-scaled information retrieval tasks, where there are thousands of relevant documents with respect to a specific query. Few users will read all of the returned documents, instead, only the top k are going to be read by users.

- **Recall at *k*** (R@k): This is the proportion of the candidates that are relevant to the query that are successfully retrieved in the top-k list. For triaging systems (e.g. in a question answering system, the relevant passages for a query is very sparse), a high R@k is desired since a high recall implies more relevant samples in the retrieved candidates, thus benefitting downstream rerankers.

- **Success at *k*** (S@k): This measures whether at least 1 relevant query is retrieved for a given query. When averaged across queries in a test set, it serves as a kind of upper bound for downstream models: if there is none retrieved in the triaging phase, no downstream model can predict the correct result.

- **Mean average precision** (MAP): For a retrieval system that returns a ranked sequence of candidates, it is desirable to consider to order in which the retrieved candidates are returned. Average precision is the average value of precision over the interval of recall $r \in [0, 1]$: AP $= \int_0^1 p(r)\mathrm{d}r$. In practice, this integral is approximated by a sum $\sum_{i=1}^{k} p(i)\Delta r(i)$, where $p(i)$ is the precision at cut-off k in the retrieved list, and $\Delta r(i)$ is the change in recall from items $i-1$ to i. Thi sum is in turn equivalent to the following formula, which is used for actual computation:

$$\mathrm{AP} = \frac{1}{|C|} \sum_{i=1}^{k} p(i)\mathrm{rel}(i) \ , \tag{2.16}$$

where $|C|$ is the number of relevant candidates, and $\mathrm{rel}(i) \in \{0, 1\}$ indicates whether the i-th returned candidate is relevant.

- **Binary preference** (b-pref): Buckley and Voorhees (2004) proposed the b-pref measure to address the problem of MAP not being stable under substantially incomplete relevance judgments. Since the candidate sets are extremely large in lots of modern applications, obtaining exhaustive annotation of relevance on the candidate set with respect to a query is infeasible. B-pref, instead, only uses binary relevance judgments to define the preference relation (any relevant candidate is preferred over any non-relevant candidate for a given query Buckley and Voorhees (2004)), and measures how frequently relevant candidates are retrieved before non-relevant documents:

$$\text{b-pref} = \frac{1}{|R|} \sum_{r \in R} 1 - \frac{h(r)}{|R|} \ , \tag{2.17}$$

where R is the non-exhaustively annotated relevant candidate set for a query, and $h(r)$ is the number of non-relevant candidates retrieved above candidate r.

These metrics can easily be computed using the standard `trec_eval` toolkit[3] developed by the TREC community, and will be discussed in detail along with their use cases in the remaining chapters.

[3] `https://github.com/usnistgov/trec_eval`.

Part II

Sentences

Chapter 3

Uncertain Inference

In this chapter, we extend the traditional categorical classification task of natural language inference to a scalar prediction task, namely *uncertain natural language inference* (UNLI), where the model is asked to predict the human subjective probability of the hypothesis happening given a premise. We explore various modeling techniques, such as regression and ranking to the problem, illustrating modeling ability to predict subjective probabilities that correlates well with humans, and finding that a ranking-based approach is better at capturing uncertainty in natural language inference. This is a ranking problem where both the queries and the candidates are natural language sentences, and the relevance function being a probability measuring textual entailment.

Some material in this chapter has been published in T. Chen, Jiang, et al. (2020). Section 3.3 are contributions of coauthors from the aforementioned article that serves as foundations to later models, hence does not form a part of this thesis' contributions.

3.1 Introduction

Various entailment tasks have been used in the natural language processing community for benchmarking systems' capability of natural language understanding. The FraCaS consortium offered the task as an evaluation mechanism, along with a small challenge set (R. Cooper, Crouch, et al., 1996), which was followed by the *recognizing textual entailment* (RTE) or *natural language inference* (NLI) tasks. In their first RTE-1 challenge (Dagan, Glickman, and Magnini, 2005) the paper states:

> We say that p entails h if, typically, a human reading p would infer that h is most likely true. This somewhat informal definition is based on (and assumes) common human understanding of language as well as common background knowledge.

Subsequent modeling efforts for RTE models the problem as a binary classification problem: given premise p and hypothesis h, determine whether p entails h or not, i.e., a classification problem where the label set is $\{$ENT, ¬ENT$\}$, standing for *entailment* and *non-entailment*.

This is later developed into classification problems where the set of discrete labels may be larger than what is described above, e.g. a 3-class label set $\{$ENT, NEU, CON$\}$ standing for *entailment*, *neutral*, and *contradiction*. Despite differences between these and recent NLI datasets (Marelli et al., 2014; Bowman et al., 2015; Williams, Nangia, and Bowman, 2018), NLI has remained a categorical prediction problem.

However, *entailment inference is uncertain and has a probabilistic nature* (Glickman, Dagan, and Koppel, 2005). Maintaining NLI as a categorical classification problem is not ideal since coarse categorical labels mask the uncertain and probabilistic nature of entailment inference. NLI pairs may share a coarse label, but the probabilities that the hypotheses are entailed by their corresponding premises may vary greatly (see Table 3.1). Hence, not all *contradictions are equally contradictory* and not all *entailments are equally entailed*.

To take this probabilistic nature of entailment inference into account, we proposed **uncertain natural language inference** (T. Chen, Jiang, et al., 2020), a refinement of NLI that captures more subtle distinctions in meaning by shifting away from categorical labels to the direct prediction of human subjective probability assessments. UNLI modifies the definition for RTE above as:

> We say that h has subjective probability y given p if, typically, a human reading p would infer that h has a y chance of being true. This somewhat informal definition is based on (and assumes) common human understanding of language as well as common background knowledge.

We illustrate that human-elicited probability assessments contain subtle distinctions on the likelihood of a hypothesis conditioned on a premise, and UNLI captures these distinctions far beyond categorical labels in popular NLI datasets (for examples, see Table 3.1).

We demonstrate how to elicit UNLI annotations. Using recent large-scale language model pre-training, we provide experimental results illustrating that systems can often predict UNLI judgments, but with clear gaps in understanding. We conclude that scalar an-

Premise $\rightsquigarrow$ Hypothesis	**NLI**	**UNLI**
A man in a white shirt taking a picture $\rightsquigarrow$ A man takes a picture	ENT	100%
A little boy in a striped shirt is standing behind a tree $\rightsquigarrow$ The boy is hiding outside	ENT	90%
A man is holding a bus pole near a building $\rightsquigarrow$ The man is waiting for the bus	NEU	74%
Woman reaching for food at the supermarket $\rightsquigarrow$ Woman is reaching for frozen corn at the store	NEU	0.1%
A smiling child is standnig behind a tree $\rightsquigarrow$ A man is eating a hotdog	CON	4%
Man laying on a platform outside on rocks $\rightsquigarrow$ Man takes a nap on his couch	CON	0%

Table 3.1: Probability assessments on NLI pairs. The NLI and UNLI columns respectively indicate the categorical label (from SNLI) and the subjective probability for the corresponding pair.

notation protocols should be adopted in future NLI-style dataset creation, which should enable new work in modeling a richer space of interesting inferences.

3.2 Related Work

The probabilistic nature and the uncertainty of natural language inference has been considered from a variety of perspectives. The notion of *probabilistic entailment* has been floating around in recent literature, as is first defined in Glickman, Dagan, and Koppel (2005) for lexical inference:

> We say that p probabilistically entails h (denoted as $p \rightsquigarrow h$) ... if t increases the likelihood of h being true, i.e. $P(\mathrm{Tr}_h = 1 \mid p) > P(\mathrm{Tr}_h = 1)$, ... Once

Likert Scale	Pavlick and Callison-Burch (2016)	S. Zhang, Rudinger, et al. (2017)
1	definite contradiction	impossible
2	likely contradiction	technically possible
3	neutral	plausible
4	likely entailment	likely
5	definite entailment	very likely

Table 3.2: Labels used by prior work for eliciting Likert-scaled inference annotations.

knowing that $p \rightsquigarrow h$, $P(\mathrm{Tr}_h = 1)$ serves as a probabilistic confidence value for h being true given p.[1]

Additionally, A. Lai and Hockenmaier (2017) noted how predicting the conditional probability of one phrase given another would be helpful in predicting textual entailment. Our UNLI work is an answer to this call: we proposed methods to annotate data and the model this direct probabilistic confidence.

There are other prior work (Pavlick and Callison-Burch, 2016; S. Zhang, Rudinger, et al., 2017) that sought to extend the 3-class classification to more fine-grained inference using a Likert scale from 1 to 5. These two papers elicited ordinal annotations reflecting likelihood judements, but then collapsed the annotations into coarse categorical labels for modeling. Their chosen labels are listed in Table 3.2. We could see that the labels used by S. Zhang, Rudinger, et al. (2017) are more probabilistically oriented than the set used by Pavlick and Callison-Burch (2016), and our work can be viewed as an extension to S. Zhang, Rudinger, et al. (2017): we directly elicit probabilistic scores.

To sum up, various historical textual inference tasks can be summarized in Table 3.3.

[1] Notations are slightly changed to accomodate to the notations used throughout this thesis.

Task	Input	Output
RTE-1 (Dagan, Glickman, and Magnini, 2005)	$p, h \in \mathcal{S}$	$y \in \mathcal{Y} = \{\neg\text{ENT}, \text{ENT}\}$
RTE-4 (Giampiccolo et al., 2008)	$p, h \in \mathcal{S}$	$y \in \mathcal{Y} = \{\text{CON}, \text{UNK}, \text{ENT}\}$
OCI (S. Zhang, Rudinger, et al., 2017)	$p, h \in \mathcal{S}$	$y \in \mathcal{Y} = \{1, 2, 3, 4, 5\}$
UNLI (T. Chen, Jiang, et al., 2020)	$p, h \in \mathcal{S}$	$y \in \mathcal{Y} = [0, 1]$

Table 3.3: Formulation of the family of entailment tasks.

Vulić et al. (2017) proposed **graded lexical entailment**, which is similar to our idea but applied to lexical-level inference, asking the question "to what degree x is a type of y". This could be seen as a probabilistic version of the logical form $x \rightarrow y$, given x and y are considered as semantic predicates.

Lalor, Wu, and H. Yu (2016) and Lalor, Wu, Munkhdalai, et al. (2018) tried capturing the uncertainty of each natural language inference pair by **item response theory** (IRT), showing fine-grained differences in discriminative power in each label.

Pavlick and Kwiatkowski (2019) recently argued that models should "*explicitly capture the full distribution of plausible human judgments*" as plausible human judgments cause inherent disagreements. Our concern is orthogonal to theirs, as we are interested in the uncertain and probabilistic nature of NLI. We are the first to propose a method for direct elicitation of subjective probability judgments on NLI pairs and direct prediction of these scalars, as opposed to reducing to categorical classification.

Recent work have also modeled the uncertainty of other semantic phenomena as direct scalar regression (and collected scalar versions of data for them) instead of categorical classification, e.g. factuality (Lee, Artzi, et al., 2015; Stanovsky et al., 2017; Rudinger, White, and Van Durme, 2018), and semantic proto-roles (Teichert et al., 2017). These efforts shares the same spirit as our work towards semantic modeling: taking uncertainty of natural language semantics into account.

Plausiblity tasks such as COPA (Roemmele, Bejan, and Gordon, 2011) and ROCStories (Mostafazadeh et al., 2016) ask models to choose the most probable examples given a context, capturing *relative* uncertainty between examples, but do not force a model to predict the probability of any hypothesis h given premise p. Z. Li, T. Chen, and Van Durme (2019) viewed this plausibility task as a *learning to rank* problem, where the model is trained to assign the highest scalar value to the most plausible outcome given context. Our work can be viewed as a variant to this, with the score being an explicit human probability judgement instead of a latent score that is not interpretable.

Linguists such as Van Eijck and Lappin (2012), Goodman and Lassiter (2015), R. Cooper, Dobnik, et al. (2015) and Bernardy et al. (2018) have described models for natural language semantics that introduce probabilities into the compositional, model-theoretic tradition begun by those such as Davidson (1967) and Montague (1973). These efforts propose new frameworks for modeling language, with examples for a fragment of the language, relying on manual transduction into a target meaning representation. Whereas they propose probabilistic models for interpreting language, we are concerned with illustrating the feasibility of eliciting probabilistic judgements on examples through crowdsourcing, and contrasting this with prior efforts that were restricted to limited categorical label sets.

Much work in AI (e.g., Garrette, Erk, and Mooney (2011)) proposes general language understanding systems with formal underpinnings based wholly or in part on probabilities. Here our focus is specifically on (U)NLI, as a motivating task for which we can gather data.

3.3 Data

We construct a UNLI dataset by eliciting subjective probabilities from crowdsource workers (Mechanical Turk) on presented premise-hypothesis pairs: Annotators are asked to estimate how likely the situation described in the hypothesis sentence would be true given the premise.

No new NLI premise-hypothesis pairs are elicited or generated, as our focus is on the *uncertainty* aspect of NLI. Owing to its familiarity within the community, we choose to illustrate UNLI via re-annotating a sampled subset of SNLI (Bowman et al., 2015) across the three categories CON / NEU / ENT.

For examples taken across the three categories CON / NEU / ENT we elicit a probability annotation $y \in [0, 1]$, resulting in what we will call *u-SNLI* (Uncertain SNLI).

We preferred SNLI over MultiNLI (Williams, Nangia, and Bowman, 2018) for this work owing to SNLI containing a subset of examples for which multiple NEU hypotheses were collected per premise. Pavlick and Kwiatkowski (2019) reported a wide range of ordinal likelihood judgments collected across SNLI NEU examples, and so we anticipated these multi-neutral premise examples to be good fodder for illustrating our points here.

There are 7,931 distinct premises in the training set of SNLI that are paired with 5 or

Premise $\rightsquigarrow$ Hypothesis	**SNLI**	**U-SNLI**
A man is singing into a microphone.		
$\rightsquigarrow$ A man performs a song.	NEU	0.946
$\rightsquigarrow$ A man is performing on stage.	NEU	0.840
$\rightsquigarrow$ A male performer is singing a special and meaningful song.	NEU	0.152
$\rightsquigarrow$ A man performing in a bar.	NEU	0.144
$\rightsquigarrow$ A man is singing the national anthem at a crowded stadium.	NEU	6.18×10^{-3}

Table 3.4: A premise in SNLI train, whose 5 hypotheses are annotated with subjective probabilities in u-SNLI.

more distinct NEU hypotheses: we take these 5 for each premise in this subset as prompts in elicitation (We believe that this is the subset of SNLI that is the hardest in terms of uncertainty – it contains the most nuanced and subtle differences between the annotations), resulting in 39,655 NEU pairs, with additional 15,862 CON and ENT pairs combined. Altogether we call this our training set, with 55,517 pairs containing 7,931 distinct premises. One such training example is shown in Table 3.4. Dev and test sets were sampled from SNLI dev and test respectively, again with heavy emphasis on NEU examples (see Table 3.5).

3.3.1 Annotation

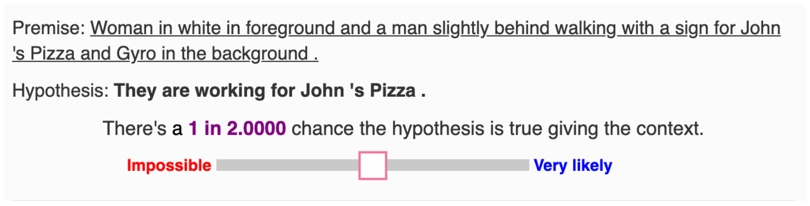
Our data annotation process was inspired by the *Efficient Annotation of Scalar Labels* (EASL) framework of Sakaguchi and Van Durme (2018), which combines notions of direct and relative assessments into a single crowd-sourcing interface. Groups of items are put into lists of size k, where k such items are presented to a user in a single page view, each item paired with a slider bar (for example, one may present $k = 5$ distinct items on one page

Partition	Breakdown	SNLI	U-SNLI
train	Distinct premises	151k	7,931
	ENT hypotheses	183k	7,931
	NEU hypotheses	183k	39,655
	CON hypotheses	183k	7,931
	Total P-H pairs	550k	55,517
dev	Distinct premises	3,319	2,647
	ENT hypotheses	3,329	162
	NEU hypotheses	3,235	2,764
	CON hypotheses	3,278	114
	Total P-H pairs	10k	3,040
test	Distinct premises	3,323	2,635
	ENT hypotheses	3,368	156
	NEU hypotheses	3,219	2,770
	CON hypotheses	3,237	114
	Total P-H pairs	10k	3,040

Table 3.5: Statistics of our u-SNLI dataset.

view). The slider bar enables direct assessment by the annotator per item. The interface has an implicit relative assessment aspect in that performing direct assessment judgments of multiple items placed visually together in a single page view is meant to encourage cross-item calibration of judgments. Our individual items were premise hypothesis pairs, with instructions requesting a probability assessment (see Figure 3.1).

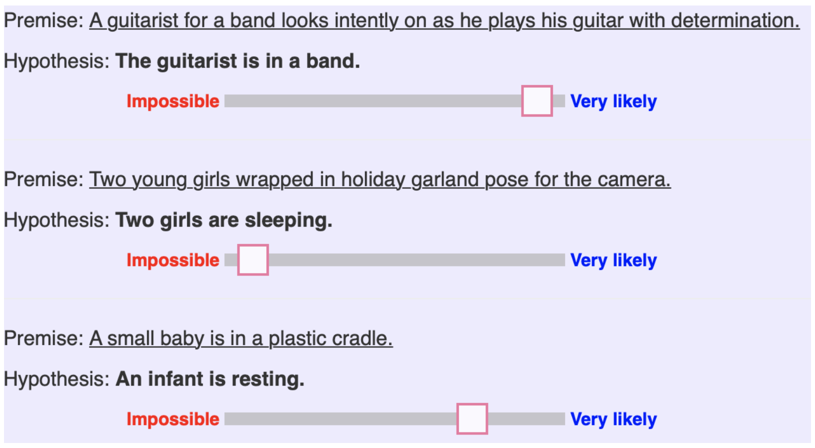
Annotators were asked to estimate how likely the situation described in the hypothesis sentence would be true given the premise. Example pairs were provided in the instructions along with suggested probability values (see Figure 3.2 for three such examples). Annotators were recommended to calibrate their score for a given element taking into account the scores provided to other elements in the same page view.


Premise: Woman in white in foreground and a man slightly behind walking with a sign for John 's Pizza and Gyro in the background .

Hypothesis: **They are working for John 's Pizza .**

There's a **1 in 2.0000** chance the hypothesis is true giving the context.

Figure 3.1: An example of our annotation interface.


Premise: A guitarist for a band looks intently on as he plays his guitar with determination.

Hypothesis: **The guitarist is in a band.**

Premise: Two young girls wrapped in holiday garland pose for the camera.

Hypothesis: **Two girls are sleeping.**

Premise: A small baby is in a plastic cradle.

Hypothesis: **An infant is resting.**

Figure 3.2: Three examples from the instructions.

Interface For each premise-hypothesis pair, we elicit a probability assessment in the interval $[0, 1]$ from annotators using an interface shown in Figure 3.1, in contrast to the uniform $\{1, \cdots, 100\}$ scale employed in the original EASL protocol. We modify the interface to allow finer-grained values near 0.0 and 1.0, following findings that humans are especially sensitive to values near the ends of the probability spectrum (Tversky and Kahneman, 1981).[2] We also use a more casual "1 in x" probability representation instead of the percentile representation for probability less than 0.5, as our pilot experiments suggested

[2]This is called the *certainty effect*: more sensitivity to the difference between, e.g., 0% and 1% than 50% and 51%.

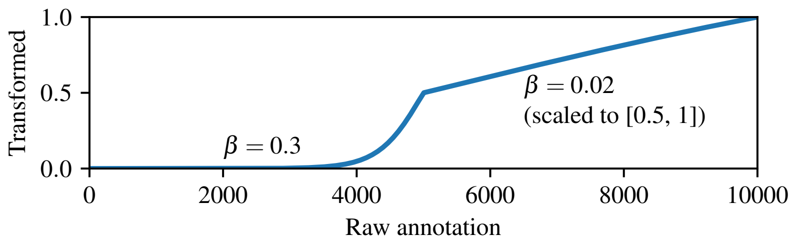


Figure 3.3: Our logistic transformation function.

this would lead to more reliable annotations near 0.0. Annotators were presented a numeric value based on a non-linear projection of the slider position ($x \in \{0, \cdots, 10000\}$):

$$f(x) = \frac{1}{1 + \exp \frac{-\beta(x-5000)}{100}} \tag{3.1}$$

We ran pilots to tune β, finding that people often choose far lower probabilities for some events than was intuitive upon inspection, (e.g., just below 50%). Therefore, we employed different β values depending on the range of $[0, 0.5]$ or $(0.5, 1]$ (Figure 3.3). Only the transformed probability score is revealed in the annotation interface, hence we are still eliciting probability judgments directly from annotators.

Qualification Test Annotators were given a qualification test to ensure non-expert workers were able to give reasonable subjective probability estimates. We first extracted seven statements from *Book of Odds* (Shapiro, Campbell, and Wright, 2014), and manually split the statement into a bleached premise and hypothesis. We then wrote three easy premise-hypothesis pairs with definite probabilities like (p = "*A girl tossed a coin.*", h = "*The coin comes up a head.*", probability: 0.5). We qualify users that meet both criteria: (1) For the three easy pairs, their annotations had to fall within a small error range around the correct label y, computed as $\delta = \frac{1}{4}\min\{y, 1-y\}$. (2) Their overall annotations have a Pearson $r > 0.7$ and Spearman $\rho > 0.4$. This qualification test led to a pool of 40 trusted annotators, which were employed for the entirety of our dataset creation.

Incremental Annotation Sakaguchi and Van Durme (2018) explored iterative annotation strategies where similar valued items from previous rounds were more likely to be placed together in subsequent rounds, meant to encourage increasingly more fine grain distinctions from annotators.

Each item was doubly annotated. In the case where the difference between the first two annotations on the raw slider bar $\{0, \cdots, 10000\}$ was greater than 2000, we elicited a third round of annotation. After annotation, the associated probability to a pair was the median of gathered responses.

Many NEU items in the SNLI can be roughly seen as sampling from a very large condition set therefore should have small probabilities near 0. However, the beta-distribution based back-end of original EASL discourages this as it tends to push labels back to the middle of its bounded support. Thus we replaced EASL's Bayesian Updating back-end with median calculation (instead of mean which is numerically identical to EASL's score response, as median is more stable when annotation number is small). Also, we select items to be re-annotated according to minimum difference between previous annotations instead of the distribution variance based criteria of original EASL as it is over-selecting items with probability in the middle. To be specific, in each round for any premise-hypothesis pair with N existing annotations corresponding to raw slider bar positions $\{p_1, p_2, \cdots, p_N\}$, we use the minimum step difference $\min_{1 \le i < j \le N} |p_i - p_j|$ as the uncertainty measure of an element. For elements with the largest k uncertainty, another additional round of annotation is performed. In our implementation we first do two rounds of annotation, therefore for each pair at most 3 annotations are elicited. The median of these 2 or 3 is chosen as the final score.

3.3.2 Data Analysis

We plot the resultant median and quartile for each of the 3 categories of SNLI under our U-SNLI training set (Figure 3.4), showing the wide range of probability judgments elicited.

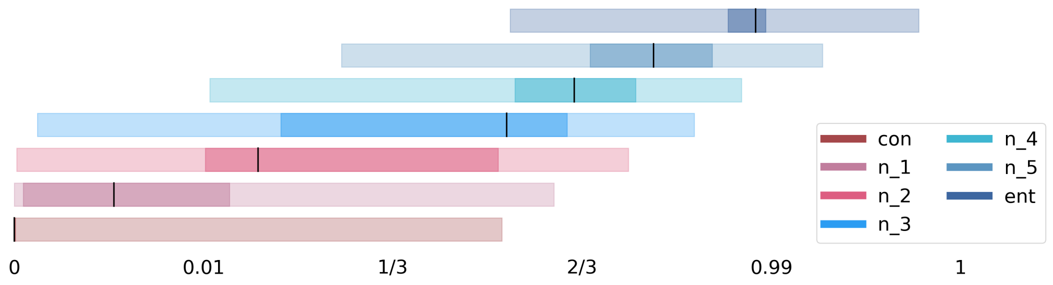


Figure 3.4: Distribution of U-SNLI training set, illustrating median and quartile for each of the 7 categories (ENT / NEU$_{1:5}$ / CON) under our scalar probability scheme. NEU$_i$ denotes the set of NEU samples labeled as the i-th least likely among the 5 hypotheses paired with each premise. Light / dark shade covers 96% / 50% of each category.

Our labels show a reasonable gradation among element pairs. In some examples, the subjective probability assessments suggest common sense based inference, e.g., "*A woman* *is at the beach*" has a 50% chance of being true knowing that "*A person is at the beach while the sun sets*."

3.4 Models for UNLI

Formally, given a premise $p \in \mathcal{S}$ and a hypothesis $h \in \mathcal{S}$, a UNLI model $F : \mathcal{S} \times \mathcal{S} \to [0, 1]$ should output an uncertainty score $\hat{y} \in [0, 1]$ of the premise-hypothesis pair that correlates well with a human-provided subjective probability assessment. This is the relevance function between premises and hypotheses.

We train regression UNLI models to predict the probability that a premise entails a hypothesis, by modifying and extending existing neural architectures used for the traditional categorical NLI. Specifically, we extend the sentence pair classifier[3] in BERT (Devlin et al., 2019), exploiting the advantages brought by large-scale language model pre-training.

We utilize the BERT model for sentence-pair classification, to exploit recent advancements in large-scale language model pre-training. This original model for categorical NLI first concatenates the premise and the hypothesis, with a special sentinel token (CLS) inserted at the beginning and a separator (SEP) inserted after each sentence. After passing this concatenated token sequence to the BERT encoder, take the encoding of the first (0th)

[3] The neural architecture for MultiNLI (Williams, Nangia, and Bowman, 2018) in BERT (Devlin et al., 2019).

sentinel (CLS) token,

$$\mathbf{f}(p,h) = \text{BERT}(\text{CLS} \; ; \; p \; ; \; \text{SEP} \; ; \; h \; ; \; \text{SEP})[0] \; , \tag{3.2}$$

and pass the resulting feature vector $\mathbf{f}(p,h)$ through a linear layer to result in the 3 classes for NLI.

We modify this structure to accommodate our scenario: we change the last layer of the network from the 3-dimensional output to a *scalar* output – the logit score $\hat{y}$. A sigmoid function σ is used as the last layer (so that the output lies in $[0,1]$, as any probability should).

$$F(p,h) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{f}(p,h)). \tag{3.3}$$

3.4.1 Regression Model

The model is trained with logistic regression instead of a cross-entropy classification loss:[4]

$$L_{\log}(\hat{y},y) = y \log \sigma(\hat{y}) + (1-y)\log(1-\sigma(\hat{y})) \; , \tag{3.4}$$

where y is the gold probabilistic judgement from the training set.

[4]We also tried the *linear* setting, where we directly use the output of the last layers as the prediction to the slider bar location and train using L_2 loss. It yielded no significant differences.

3.4.2 Ranking Model

Since we focus on the *uncertainty* of NLI, we alternatively approach the problem as a *learning to rank* problem. Instead of regression, we train a model that could correctly *rank* the premise-hypothesis pairs according to the probability: ENT $\succ$ NEU $\succ$ CON. To this end, we train the UNLI model $F : \mathcal{S} \times \mathcal{S} \to [0, 1]$ with a margin-based loss (Weston and Watkins, 1999):

$$\sum_{y_i \succ y_j} \max\{0, \xi - F(p_i, h_i) + F(p_j, h_j)\} , \tag{3.5}$$

where ξ is a *constant margin* hyperparameter. This is to say, the model learns to assign a higher score for $F(p, h)$ than $F(p', h')$ if $t_i \succ t_j$, ideally the gap being larger than ξ. We also experiment with a *linear* margin that is proportional to their difference:

$$\sum_{y_i \succ y_j} \max\{0, \xi \cdot (y_i - y_j) - F(p_i, h_i) + F(p_j, h_j)\} , \tag{3.6}$$

where the margin $\xi \cdot (y_i - y_j)$ depends on the label. We term this as the *linear margin* case, and can be seen as a special case of structured SVMs (Tsochantaridis et al., 2005).

Preference Pairs However, the summation in Equation 3.5 is over $R = \{(i, j) \mid y_i \succ y_j\}$, which unfortunately has a computationally infeasible $\mathcal{O}(N^2)$ complexity, where N is the number of samples in the dataset. Owing to the discussion in Chapter 2 about preference pairs, we take the summation over subsets:

- **(1) Shared-premise pairs**: data pairs with identical premises are included — these pairs rank the probability of different hypotheses given the same premise:

$$R_1 = \{(i, j) \mid p_i = p_j \wedge y_i \succ y_j\}; \tag{3.7}$$

- **(2) Cross-premise pairs**: For each sample i, we randomly sample K other samples S_i with different premises and lower probability: [5]

$$R_2 = \{(i, j) \mid j \in S_i \wedge y_i \succ y_j\}. \tag{3.8}$$

The union $R_1 \cup R_2$ is used as training set, hence reducing the complexity to $\mathcal{O}(KN)$. Our experiments found that $K = 1$ is better than no cross-premise pairs, but $K = 2$ does not lead to further improvement.

3.4.3 Metrics

We compute Pearson correlation (r), the Spearman rank correlation (ρ), and the mean square error (MSE) between y and $\hat{y}$ as the metrics to measure the performance of UNLI models. Pearson r measures the linear correlation between the gold probability assessments and model's output; Spearman ρ measures the ability of the model ranking the premise-hypothesis pairs with respect to their subjective probability; MSE measures whether the model can recover the subjective probability value from premise-hypothesis pairs. A high r and ρ, but a low MSE is desired.

[5] We skip this for CON samples in SNLI since there are no samples with lower probability.

3.5 Results & Analyses

3.5.1 Hypothesis-only baselines

Owing to the concerns raised with *annotation artifacts* in SNLI (Gururangan et al., 2018; Poliak et al., 2018; Tsuchiya, 2018), we include a *hypothesis-only baseline* that only encodes hypotheses.

Under our formulation of UNLI, the *hypothesis-only baseline* takes a more interesting interpretation: given hypothesis h, the hypothesis-only UNLI score $F(\emptyset, h)$ can be considered as the subjectively probability of h happening regardless of context: with an abuse of notation, we could say that this is essentially *marginalizing out* all possible premises.

Table 3.6 reports results on u-SNLI dev and test sets. Just training on 55,517 u-SNLI examples yields a 62.71% Pearson r on test. The hypothesis-only baseline achieved a correlation around 40%. This result corroborates the findings that a hidden bias exists in the SNLI dataset's hypotheses, and shows this bias may also exist in u-SNLI. This is unsurprising because u-SNLI examples are sampled from SNLI.

	Hyp-only		**Full-model**	
	Dev	**Test**	**Dev**	**Test**
$\boldsymbol{r}$	0.376	0.412	0.638	0.627
$\boldsymbol{\rho}$	0.385	0.417	0.641	0.630
MSE	0.109	0.106	0.075	0.078

Table 3.6: Metrics for training on u-SNLI.

3.5.2 Human Performance

We elicit additional annotations on *u*-SNLI dev set to establish a randomly sampled human performance. We use the same annotators as before but ensure each annotator has not previously seen the pair they are annotating. We average the scores from three-way redundant elicitation. This setting approximates the performance of a randomly sampled human on *u*-SNLI, and is therefore a reasonable lower bound on the performance one could achieve with a dedicated, trained single human annotator. Under this setting it yields $r = 0.6978$, $\rho = 0.7273$, and MSE = 0.0759: our regression model trained on *u*-SNLI is therefore approaching human performance. While encouraging, the model fails drastically for some examples.

3.5.3 Qualitative Error Analysis

Table 3.7 illustrates examples with large gaps between the gold probability assessment and the BERT-based model output. The model seems to have learned lexicon-level inference (e.g., *race cars* $\rightsquigarrow$ *going fast*, but ignored crucial information (*sits in the pits*), and fails to learn certain commonsense patterns (e.g. *riding amusement park ride* $\rightsquigarrow$ *screaming*; *man and woman drinking at a bar* $\rightsquigarrow$ *on a date*). These examples illustrate the model's insufficient commonsense reasoning and plausibility estimation.

Premise $\rightsquigarrow$ Hypothesis	**SNLI**	***u*-SNLI**	**Predicted**
A man perched on a row of aquariums is using a net to scoop a fish from another aquarium. $\rightsquigarrow$ A man is standing by the aquariums.	ENT	1.0	0.119
A man and woman are drinking at a bar. $\rightsquigarrow$ A couple is out on a date.	NEU	0.755	0.377
Couple walking on the beach. $\rightsquigarrow$ The couple are holding hands.	NEU	0.808	0.308
An elderly woman crafts a design on a loom. $\rightsquigarrow$ The woman is a seamstress.	NEU	0.923	0.197
Two girls riding an amusement park ride. $\rightsquigarrow$ The two girls are screaming.	NEU	0.909	0.075
A man and woman sit at a cluttered table. $\rightsquigarrow$ The table is neat and clean.	CON	4.91×10^{-4}	0.262
A race car sits in the pits. $\rightsquigarrow$ The car is going fast.	CON	2.88×10^{-7}	0.724
A guy is standing in front of a toilet with a coffee cup in one hand and a toilet brush in the other. $\rightsquigarrow$ A man is attempting to brew coffee.	CON	8.32×10^{-6}	0.504

Table 3.7: Selected *u*-SNLI dev examples where BERT predictions greatly deviate from gold assessments.

3.5.4 Pre-training with SNLI

Can we leverage the remaining roughly 500,000 SNLI training pairs that only have categorical labels? We devise two pre-training methods.

Surrogate Regression One method would be to train a categorical NLI model on SNLI and when fine-tuning on *u*-SNLI, replace the last layer of the network from a categorical prediction with a sigmoid function.[6] However, a typical categorical loss function would not

[6] This is similar to how **babies** pre-train on SNLI, then fine-tune the model using their *Add-One* pairs.

take into account the ordering between the different categorical labels.[7] Instead, we derive a surrogate function $s : \mathcal{T} \to [0, 1]$ that maps SNLI categorical labels $t \in \{\text{ENT}, \text{NEU}, \text{CON}\}$ to the average score of all u-SNLI training annotations labeled with t in SNLI.[8]

We use this mapping to pre-train a regression model on the SNLI training examples not included in u-SNLI. We also fine-tune the model on u-SNLI's training set. Table 3.6 reports the results evaluated on u-SNLI's dev and test sets. The model trained on the roughly $500K$ mapped SNLI examples, performs much worse than when trained on just about $55K$ u-SNLI examples.

Ranking on SNLI As we have noted before, under our entailment relevance function, we should have ENT $\succ$ NEU $\succ$ CON. Hence we could directly derive a ranking preference pair set from SNLI. Given the SNLI dataset $\text{SNLI} = \{(p, h, y)\}$ where $p, h \in \mathcal{S}$ and $y \in \{\text{ENT}, \text{NEU}, \text{CON}\}$, we construct the preference pair set:

$$\mathcal{R}_{\text{SNLI}} = \{(p, h_+, h_-) \mid (p, h_+, y_+), (p, h_-, y_-) \in \text{SNLI}, y_+ \succ y_-\} , \tag{3.9}$$

then apply a pairwise ranking loss:

$$L = \sum_{(p,h_+,h_-)\in\mathcal{R}_{\text{SNLI}}} \max\{0, \xi_{\text{pretrain}} - F(p, h_+) + F(p, h_-)\} . \tag{3.10}$$

Note that the margin hyperparameter ξ_{pretrain} is different from the linear margin ξ in Equa-

[7] That the score of ENT > score of NEU > score of CON.
[8] $s : \{\text{ENT} \mapsto 0.9272; \text{NEU} \mapsto 0.4250; \text{CON} \mapsto 0.0209\}$.

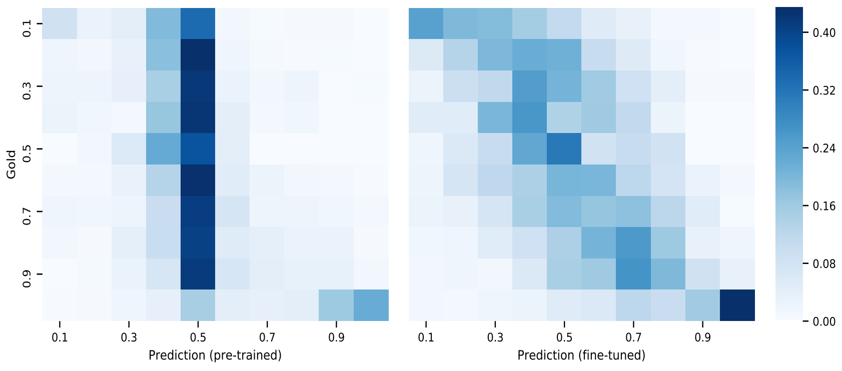
Setting	**Dev**			**Test**		
	r	***ρ***	**MSE**	***r***	***ρ***	**MSE**
SNLI (Surrogate regression)	0.520	0.524	0.093	0.496	0.523	0.097
SNLI (Ranking)	0.545	0.533	0.093	0.539	0.538	0.095
u-SNLI (Ranking)	0.623	0.634	0.087	0.611	0.623	0.089
u-SNLI (Regression)	0.638	0.641	0.075	0.627	0.635	0.078
+ SNLI regression pre-training	0.676	0.681	0.069	0.659	0.671	0.073
+ SNLI ranking pre-training	**0.680**	**0.683**	**0.068**	**0.665**	**0.675**	**0.072**

Table 3.8: Metrics of the prediction models under various configurations for *u*-SNLI.

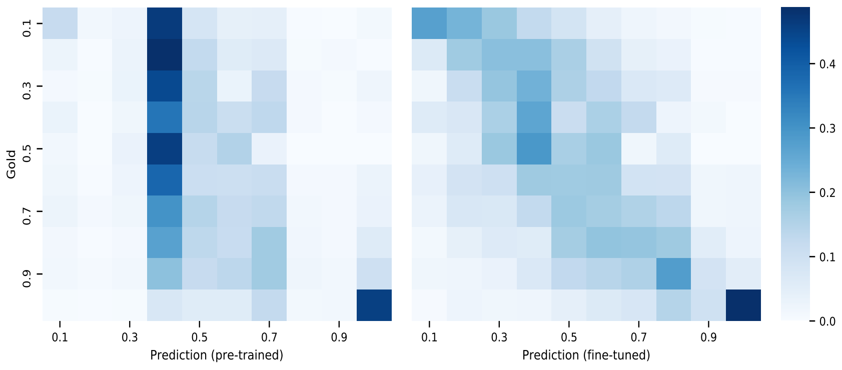
tion 3.5 or Equation 3.6: This ξ is for pre-training purposes, and different from what is used for fine-tuning on *u*-SNLI.

When we pre-train the model on the SNLI using either of the two pre-training methods (surrogate regression and ranking) and fine-tune on *u*-SNLI, results noticeably improve (see Table 3.8). This improvement is akin to the Phang, Févry, and Bowman (2018)'s finding that many NLI datasets cover informative signal for different tasks, explaining why pre-training on NLI can be advantageous. Here, an impoverished version of UNLI is helpful.

Additionally, in Table 3.8, we could find that when training on *u*-SNLI alone, ranking and regression produces similar Pearson or Spearman coefficients, but the actual value measured under MSE is better under a regression loss (0.078 < 0.089). However, for SNLI, ranking produces better correlation with human judgements than the surrogate regression (r: 0.539 > 0.496): Since we only know the ranking between hypotheses in UNLI but do not know the actual the relevance scores of SNLI inference pairs, using a pairwise ranking loss is the more suitable choice. When ranking on SNLI is used as pre-training, it also leads to slight performance improvements even after fine-tuning on *u*-SNLI (r: 0.665 > 0.659).



(a) Left: SNLI surrogate regression, r = 51.98%;
Right: After fine-tuning with U-SNLI, r = 67.62%.



(b) Left: SNLI learning to rank, r = 54.51%;
Right: After fine-tuning with U-SNLI, r = 67.97%.

Figure 3.5: Heatmap of the predictions on U-SNLI dev set under the pretrained (left) and the fine-tuned (right); regression pre-trained (top) and ranking pre-trained (bottom) models. Prediction frequencies are normalized along each gold label row.

3.5.5 Model behavior

Figure 3.5 depicts the model behavior when training just on SNLI or fine-tuning with *u*-SNLI. When using the original SNLI data, under the surrogate regression setting, the model's prediction concentrates on the 3 surrogate scalar values of the 3 SNLI classes. After fine-tuning on *u*-SNLI, the model learns smoother predictions for premise-hypothesis pairs, supported by the superior Pearson correlation score. The darker boxes in bottom-

Preteen girl with blond-hair plays with bubbles near a vendor stall in a mall courtyard. $\rightsquigarrow_{0.162}$ *The girl is* ***ten***. Alternatives: a newborn | one | two | three | $\cdots$ | ten | eleven | twelve
Three young men standing in a field behind a barbecue smiling each giving the two handed thumbs up sign. $\rightsquigarrow_{0.525}$ *Three men are barbecuing* ***lunch***. Alternatives: breakfast | lunch | dinner

Table 3.9: Examples from SNLI prompting a question of logical coherence of crowd-sourced probabilities.

right corner of the heatmaps indicate high accuracy on samples with $\approx$ 1.0 gold *u*-SNLI labels and $\approx$ 1.0 model predictions, signifying that our UNLI models are very good at recognizing entailments.

3.5.6 A Case Study on Coherence

We defined UNLI following RTE, in terms of human responses to h given p. Are these responses logically coherent when taken over a larger space of hypotheses? We are eager to consider future work that explores the connections between predicting crowdsourced annotations on NLI-inspired items, with the rich study of epistemology and uncertainty in philosophy and psychology. Here we limit ourselves to two cases (see Table 3.9) that probe the logical coherence of our crowd-elicited probabilities.

Both examples were selected from SNLI train owing to the premise establishing the potential for a common-sense, finitely enumerable set of alternatives. We manually constructed alternatives such that they were: (1) logically mutually exclusive; and (2) one of the hypotheses must reasonably hold given the premise. Specifically, a *preteen* must have

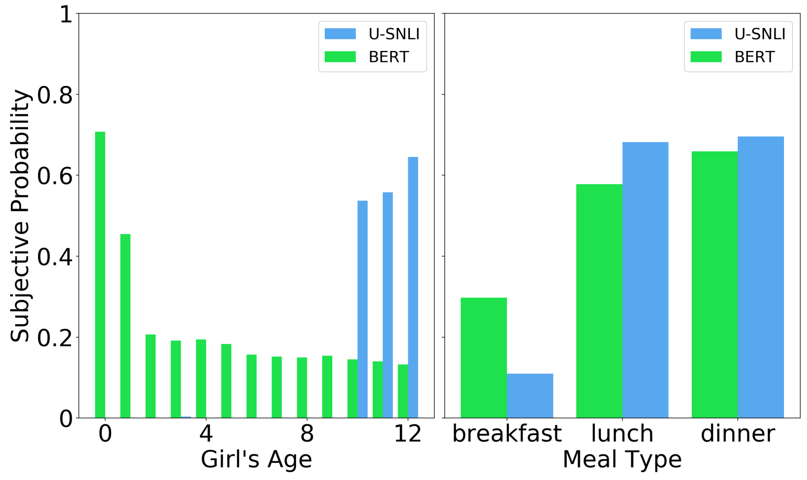


Figure 3.6: Subjective probability for *the preteen girl* (left) and *the barbecued meal* (right).

an age in the range of 0 ... 12, and the most common-sense alternatives to *lunch* include *breakfast* and *dinner*. We distribute these constructed pairs into separate HITs making sure that no annotator is viewing two related premise-hypothesis pairs at the same time, employing 6-way redundancy. We observe that the sum of resultant scores exceeds 1.0 in both cases. That humans can be irrational in their probability assignments is well known, and therefore this result is neither a flaw nor unexpected: in UNLI we have embraced human judgments, taking seriously the phrasing of the original RTE task. Future works may consider how strongly we wish Natural Language Understanding (NLU) models to reproduce human incoherence.

3.6 Conclusion

We proposed **Uncertain Natural Language Inference** (UNLI), a new task of directly predicting human likelihood judgments on NLI premise-hypothesis pairs. In short, we have shown that not all NLI contradictions are created equal, nor neutrals, nor entailments. We demonstrated that (1) eliciting supporting data is feasible, and (2) annotations in the data can be used for improving a scalar regression model beyond the information contained in existing categorical labels, using recent contextualized word embeddings, e.g. BERT.

We proposed models based on regression and ranking, and utilized ranking methods on the coarser-annotated traditional NLI datasets (e.g. SNLI) as pre-training, leading to improvements on UNLI model performance.

Humans are able to make finer distinctions between meanings than is being captured by current annotation approaches; we advocate the community strives for systems that can do the same, and therefore shift away from categorical NLI labels and move to something more fine-grained such as our UNLI protocol.

This concludes the discussion of ranking in the context of natural language inference where we take the uncertainty nature of human language into account. In the next chapter, we turn our focus to ranking short documents with respect to how likely they evoke *situation frames*.

Chapter 4

Predicting Situation Frames

In this chapter we switch the ranking target from another text snippet (as in UNLI) to a label: ranking the relevance a text snippet to a specific label in a designated label set. We will focus on the task of predicting *situation frames*, and show how ranking methods can be used for these types of problems. This enables NLP systems' capability to respond to incidents like a natural disaster, or a regime change, within a short time of the emergence of that incident, especially under low-resource language scenarios.

Some materials presented in this chapter can be found in the non-peer-reviewed archive from NIST LoReHLT19 evaluation program (M. Zhang et al., 2019). Section 4.3 contains background material contributed by other performers in the LoReHLT19 evaluation program, hence should not be considered as contribution of this thesis.

4.1 Introduction

Most of the world's languages are under-resourced for developing human language technologies, but the lack of linguistic resources does not correlate with the lack of need for such technologies (Rehms and Uszkoreit, 2012). The DARPA Low Resource Languages for Emerging Incidents (LORELEI) program aims to advance the technologies of NLP in these low-resource languages. The program is especially concerned with the ability to obtain *situational awareness* (Strassel and Tracey, 2016; Strassel, Bies, and Tracey, 2017): How can an NLP system respond to incidents like a natural disaster, or a regime change, within a short time of the emergence of that incident, under the scenario where the information obtained are all in low-resource languages?

This need leads to the development of an evaluation program called *Situation Frames* (SF). SF data labels basic information relevant to humanitarian aid and disaster relief (HADR) scenarios, so that systems can respond to actional information contained in these HADR-related documents. Mission planners might require these information in order to mount a quick response to these types of incidents (Griffitt et al., 2018). There are 11 SF types defined in the evaluation program (see below) that serve as prediction targets.

To test the performance of these situation frame detection systems, the evaluation program devised a schema where training data and test data do not overlap in languages: The Representative Language Packs, containing large volumes of formal and informal text with annotations to support situational awareness, are used as training datasets (e.g. English, Bengali, Farsi, Amharic, etc.), whereas the Incident Languages (IL) Packs, containing manually labeled evaluation data designed to test system performance, are in one or more *surprise* languages (e.g. Oromo, Ukrainian, Uyghur) that remain unknown until the start of the annual evaluation (Strassel and Tracey, 2016).

No training data is provided for incident languages (ILs), since in the actual use of these SF systems, data is unlikely to be available at the start of an incident involving a low resource language. Systems must utilize techniques such as transfer learning and language universals in order to rapidly respond to the need for situational awareness in a new language (Griffitt et al., 2018).

In this section, the problem of SF prediction is cast as a multi-label regression problem as opposed to conventional document classification: for each text snippet x, and for each SF type y, the model outputs a confidence score $F(x, y)$ that is representative of how much the text snippet x evokes a situation frame of type y.

We approach this with a *learning-to-rank* method, to synergize with our scalar annotations. Scalar annotations on crowdsourcing platforms such as Amazon Mechanical Turk, instead of categorical annotations, enables eliciting *graded* response that are crucial in many NLP tasks (Sakaguchi and Van Durme, 2018). Since annotators may not have a pre-existing or well-calibrated scale for the annotation of a specific task, we explore models that take into account *per-worker* or *per-HIT* (Human Intelligence Task) biases via learning-to-rank, leading to better performance on the SF prediction task.

4.2 Task Definition

Given a collection of documents (in the form of text snippets), in the incident language (IL) or English, a situation frame detection (SF) system is required to automatically identify zero or more situation frames in the text snippet.

The type of situation frames, as defined in the NIST LoReHLT 2019 evaluation,[1] consists of the following *need* frame types. These frame types are defined by the LORELEI project and from existing annotation schemes, i.e. MicroMappers (Imran et al., 2014).

- `evac`: Evacuation
- `food`: Food supply
- `infra`: Infrastructure
- `med`: Medical assistance
- `search`: Search/rescue
- `shelter`: shelter
- `utils`: Utilities/energy/sanitation
- `water`: Water supply

And the following *issue* frames:

- `regimechange`: Regime change
- `crimeviolence`: Civil unrest / widespread crime
- `terrorism`: Terrorism / extreme violence

Multiple needs or issues in a document leads to mulitple frames labeled as positive, effectively turning the SF prediction problem to a *multi-label* prediction problem: given a

[1] `https://www.nist.gov/system/files/documents/2019/06/19/nist_lorehlt_2019_evaluation_plan_v1.0.pdf`.

document x in an incident language (IL), predict the set of possible SF types document x may evoke, and the confidence score for each type predicted.

4.3 Background Discription

4.3.1 Data Collection

We collected *scalar* annotations (on the scale from 1 to 100 that indicates how much a document evokes a specific need/issue) using the EASL annotation scheme (Sakaguchi and Van Durme, 2018). The scalar annotation compensates for noisy annotations and inconsistent thresholds between annotators or between HITs.

Sentence-level SF scalar labels are collected from the LoReHLT language pack data and additional data constructed as part of the LoReHLT evaluation program,[2] with additional tweet collections of various high-profile incidents, including:

- 2011 volcanic explosions in Eritrea
- 2011 major droughts in East Africa
- 2013 major 7.7-scale earthquake in Iran near Pakistan border
- 2013 overthrow of Morsi and replacement by el-Sissi in Egypt
- 2013 Cycone Phallin in India
- 2014 brutal crackdown on student protesters in Ethiopia
- 2011 mass flooding in Turkey
- 2015 ISIS suicide-attack & shooting in Paris

[2] This includes a tweet collection about incidents in Nepal from Douglas A. Jones of the MIT Lincoln Laboratory.

4.3.2 Embeddings

We use cross-lingual word embeddings (Artetxe, Labaka, and Agirre, 2016; Joulin et al., 2018; Lample et al., 2018) as features, since they allow models trained on English SF data to be applied to incident language (IL) documents. An off-line alignment method for trainign cross-lingual word embeddings is used.

A set of monolingual word embeddings are trained using fastText with the skip-gram setting (Mikolov et al., 2018). Incident language (IL) embeddings are trained from the provided documents and then aligned them to centered and normalized English and Bengali Wikipedia embeddings. The alignment methods used are MultiCCA (Ammar et al., 2016) and RCSLS (Joulin et al., 2018). Dictionaries used to align the embeddings include (1) Set0 category-I dictionary; (2) NI annotations; (3) aligned words extracted from Set0 parallel texts with fastAlign; (4) a Unimorph-derived dictionary. MultiCCA uses (1), (2), (3), and (5), whereas RCSLS uses (1) and (3), based on intrinsic evaluations. We use 300-dimensional cross-lingual word embeddings for further experiments and discussions.

4.3.3 Scalar Annotations

Our training data based on EASL is gathered in batches of five (each batch forms a HIT): a single page has five distinct messages, each of which are assigned a scalar value from 1 ‘not present’ to 100 ‘present’. Instead of converting these values to binary judgments for sake of training binary classification models, we explore the idea that at model training time we should not be building binary classifiers, but instead learning to *rank* content with respect to how likely they evoke a given SF type. In addition, we explore whether different crowdsource annotators may differ significantly in their annotations (*per-worker* bias), or even whether a single annotator may make use of the scalar range a little differently from page to page (each batch of five, *per-HIT* bias).

4.4 Ranking Model

We explore these ideas by training models under a pairwise loss: given two messages that were annotated under EASL, the two messages need to have derived scores whose gap is proportional to the annotated scalar difference between the two messages. Additionally, we employ a word-level attention with a "type" vector as the query (each SF is associated with such a type vector, which is learned by the system), so that salient part of the message will be attended.

Formally, given a specific SF type y, and a message $x = (w_1, \cdots, w_n)$, an attended representation $\tilde{\mathbf{x}}$ of the message is computed as follows:

$$a_i(y) \propto \exp(\mathbf{y} \cdot \mathbf{w}_i) \tag{4.1}$$

$$\tilde{\mathbf{x}}(y) = \sum_{i=1}^{n} a_i \mathbf{w}_i \tag{4.2}$$

where $\mathbf{w}_i \in \mathbb{R}^d$ is the cross-lingual word embedding of word w_i, and $\mathbf{y} \in \mathbb{R}^d$ is a learned *type* embedding of SF type. Here the dimension of the word embeddings $d = 300$.

This is similar to the span embedding method proposed by Lee, L. He, Lewis, et al. (2017): where they used a fixed query vector to attend to a mention span to get a weighted average of the word encodings in the span. Our method here is different in two ways: **(1)** instead of a global fixed query vector, our vector depends on the target prediction SF type: so that encodings salient to each SF type can be extracted separately instead of a type-agnostic embedding; **(2)** We do not have a mention or text span here: the whole text snippet is attended using the type vector.

A final score of the relevance of the document x and SF type y is obtained by combining the type-specific document embedding and the type embedding using various matching methods (Mou et al., 2016) and pass that through a 3-layer neural network. The last layer is a sigmoid layer that converts the output to a score in $[0, 1]$:

$$F(x, y) = \sigma(\mathrm{FFNN}([\tilde{\mathbf{x}}(y) \; ; \; \mathbf{y} \; ; \; \tilde{\mathbf{x}}(y) \cdot \mathbf{y} \; ; \; |\tilde{\mathbf{x}}(y) - \mathbf{y}|])) \tag{4.3}$$

Then given two messages x_1 and x_2 where x_1 is more likely to evoke SF type y, we employ a ranking loss (Weston and Watkins, 1999):

$$L(x_1, x_2, y) = \max\{0, \xi - F(x_1, y) + F(x_2, y)\} \tag{4.4}$$

This loss encourages the model to learn that message x_1 should be more likely (by a margin of ξ) to evoke type y than x_2.

For example, given a SF type "`food`", a message saying *I'm hungry* would need to be *further away* from a message like *I'm bored*, than a message like *Food is good* (questionably evoking a FOOD need, but not confidently). After training models to rank content under each SF type, we calibrated our model scores against NI annotations to determine what the final binary thresholds should be. Internal experiments suggested that training based pairs of examples taken from single batches (pairs from the same page of five examples, by the same annotator) led to the most robust model. We have remaining experiments to perform to validate if this finding is robust, but our initial results were sufficiently promising to employ this approach for some of our evaluations submissions.

4.4.1 Generation of Ranking Pairs

When using Equation 4.4 as our loss, a question arise: what (x_1, x_2) pairs do we pick as training data?

As we have discussed before, we explore *per-worker* biases and *per-HIT* biases here. Let us assume that each annotation we gathered from EASL forms a 4-tuple in the form of

$$\mathcal{D} = \{(\texttt{worker}, \texttt{hit}, x, y, s)\}\ , \tag{4.5}$$

where `worker` and `hit` are string IDs that uniquely identifies an MTurk worker or a particular instance of HIT; x is the document, y is an SF type, and $s \in [0, 1]$ is the score workers assigned to the relevance of document x pertaining to SF type y.

We experiment with the following two ways of sampling the pairs:

- **Group by workers**: Only sample documents x_1 and x_2 if they are annotated by the same worker:

$$\mathcal{R}_{\text{worker}} = \{(x_1, x_2, y) \mid s_1 > s_2 \wedge \texttt{worker}_1 = \texttt{worker}_2\} \tag{4.6}$$

  This accounts for worker biases: Worker A might be generally assigning higher scores to documents, whereas worker B may assign lower scores. By only training on these pairs, the model only compares the annotations done by the same worker. This might lead to a reduction in biases from workers.

- **Group by HIT**: Only sample documents x_1 and x_2 if they are annotated by the same worker in the same HIT:

$$\mathcal{R}_{\text{HIT}} = \{(x_1, x_2, y) \mid s_1 > s_2 \wedge \texttt{hit}_1 = \texttt{hit}_2\} \tag{4.7}$$

  Even for the same worker, they could exhibit different thresholds between HITs. We can safely assume that in the same HIT, the annotations are generally well calibrated.

4.5 Experiments

We use the two sampling method (Group by worker / group by HIT) for ranking discussed above.

The data collected are randomly split into a training set (90%) and a dev set (10%).

The training set consists of 53,296 samples, and the dev set 6,144 samples. Note that the split for the two sampling method are different: for the "group by worker" case, we ensure *no overlap* between workers in the training and dev set; similarly, for the "group by HIT" case, there is no overlap between HITs.

Metrics Similar to UNLI, since the prediction target is scalar instead of categorical, we use Pearson correlation coefficient to measure the linear correlation between the annotations and the predictions.

Results Table 4.1 shows that for most of the situation frame types, ranking models based on preference pairs in HITs are better than models trained on within-worker preference pairs. Despite the larger size of the preference pair set used in training, we hypothesize that grouping by HIT reduces inter-annotator and inter-HIT bias, leading the better correlation between human annotations and model predictions.

4.6 Conclusion

We proposed a learning-to-rank method to detect situation frames in short documents. Particularly, we explored the design choices when constructing the preference pairs for ranking: namely grouping by worker IDs, and grouping by HIT IDs. While the current results are not conclusive, we believe that these design choices affect how the model treats

SF Type	Group by worker	Group by HIT
`evac`	0.495	**0.567**
`food`	0.582	**0.649**
`infra`	0.405	**0.436**
`med`	0.640	**0.677**
`search`	0.646	**0.690**
`shelter`	0.566	**0.673**
`utils`	**0.598**	0.561
`water`	0.508	**0.574**
`regimechange`	**0.667**	0.652
`crimeviolence`	0.674	**0.732**
`terrorism`	**0.590**	0.552

Table 4.1: The results on the internal dev set on the situation frame task.

per-worker biases and per-HIT biases, since taking these biases into account result in improvement in performance of a scalar prediction task.

Future modeling on similar tasks with scalar annotations (such as sentiment analysis, sentence similarity, etc.) should consider modeling with explicit consideration of inter-annotator and inter-HIT bias, as the ranking model in this chapter suggests.

Chapter 5

Discriminative Retrieval

In this chapter, we propose a framework for discriminative information retrieval atop linguistic features, trained to improve the recall of answer candidate passage retrieval, the initial step in text-based question answering. We formalize this as an instance of linear feature-based information retrieval, demonstrating significant improvement in recall for candidate triage in question answering.

This is an instance of retrieval over sentences, where a given query is a natural language question, with the candidate set being the set of all sentences in a corpus. Our model can be considered as a bag-of-features approach, and the retrieval effectively reuses infrastructures built for bag-of-words retrieval, therefore obtaining scalability similar to traiditional IR techniques but with much richer feature representations.

Materials in this chapter have been published in T. Chen and Van Durme (2017).

5.1 Introduction

Question answering (QA) with textual corpora is typically modeled as first finding a candidate set of passages (sentences) that may contain an answer to a question, followed by an optional candidate reranking stage, and then finally an information extraction (IE) step to select the answer string (Greenwood, 2008; Yao, Van Durme, and P. Clark, 2013; D. Chen et al., 2017). QA systems normally employ an information retrieval (IR) system to produce the initial set of candidates, usually treated as a black box, bag-of-words process that selects candidate passages best overlapping with the content in the question.

Recent efforts in corpus-based QA have been focused heavily on *reranking*, or *answer sentence selection*: filtering the candidate set as a supervised classification task to single out those that answer the given question. Extensive research has explored employing syntactic or semantic hand-crafted features (Yih et al., 2013; M. Wang and Manning, 2010; Heilman and Smith, 2010; Yao, Van Durme, Callison-Burch, et al., 2013), and recently using neural networks with various architectures (L. Yu et al., 2014; Severyn and Moschitti, 2015; D. Wang and Nyberg, 2015; Yin et al., 2016).

The shared aspect of all these approaches is that the quality of reranking a candidate set is upper-bounded by the initial set of candidates: unless one plans on reranking the *entire* corpus for each question as it arrives, one is still reliant on an initial IR stage in order to obtain a computationally feasible QA system.

We propose a framework for performing this triage step for QA sentence selection and other related tasks in sublinear time. Our method shows a log-linear model can be trained to optimize an objective function for downstream reranking, and the resulting trained weights can be reused to retrieve a candidate set. The content that our method retrieves is what the downstream components are known to prefer: it is ***trainable*** using the same data as employed in training candidate reranking. Our approach follows Yao, Van Durme, and P. Clark (2013) who proposed the automatic coupling of QA sentence selection and IR by augmenting a bag-of-words query with desired named entity (NE) types based on a given question. While Yao et al. showed improved performance in IR as compared with an off-the-shelf IR system, the model was proof-of-concept, employing a simple linear interpolation between bag-of-words and NE features with a single scalar value tuned on a development set, kept static across all types of questions at test time. We generalize Yao, Van Durme, and P. Clark (2013)'s intuition by casting the problem as an instance of classification-based retrieval (Robertson and Jones, 1976), formalized as a discriminative retrieval model (W. S. Cooper, Gey, and Dabney, 1992; Gey, 1994; Nallapati, 2004) allowing for the use of NLP features. Our framework can then be viewed as an instance of linear feature-based IR, following Metzler and Croft (2007).

To implement this approach, we propose a general feature-driven abstraction for coupling retrieval and answer sentence selection. Our experiments demonstrate state-of-the-art results on QA sentence selection on the dataset of J. J. Lin and Katz (2006), and we show significant improvements over a bag-of-words of baseline on a novel Wikipedia-derived dataset we introduce here, based on WikiQA (Yang, Yih, and Meek, 2015).

5.2 Background

5.2.1 Discriminative Information Retrieval

Traditional information retrieval (IR) models viewed the retrieval problem as measuring the similarity, often the cosine similarity, or the inner product, between two bag-of-words vectors, i.e., between the query and the candidates. One of these standard functions used for ranking is the **Okapi BM25** (Robertson, Walker, et al., 1994), which is a variant of the TF-IDF function. Recalling the formulation of BM25 (Equation 2.1) in Chapter 2, we can see that the BM25 relevance function can be decomposed as an inner product of sparse vectors in the form of

$$\mathbf{f}_Q(q) = \{t : \mathrm{freq}(t, q) \cdot \mathrm{idf}(t) \mid t \in V\} \tag{5.1}$$

$$\mathbf{f}_D(d) = \left\{ t : \frac{(k_1 + 1) \cdot \mathrm{freq}(t, d)}{\mathrm{freq}(t, d) + k_1\left(1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}}\right)} \;\middle|\; t \in V \right\} \tag{5.2}$$

$$F(q, d) = \mathbf{f}_Q(q) \cdot \mathbf{f}_D(d) \tag{5.3}$$

One shortcoming of this vector-space model (VSM) is that it did not provide a theoretical basis for computing the optimum weights. The binary independence retrieval (BIR) (Robertson and Jones, 1976) viewed IR as a classification problem that classifies the entire collection of candidates into two classes: relevant and irrelevant. In the framework of BIR, a probability of $P(p,q)$ is computed and ranked to generate the retrieved list. In this view of casting IR as a discriminative model, sophisticated machine learning techniques can be leveraged.

W. S. Cooper, Gey, and Dabney (1992), Gey (1994), and Nallapati (2004) further formalized this framework into a logistic regression (log-linear) retrieval model. Another prominent example of employing discriminative models in IR is by language modeling (Ponte and Croft, 1998).

5.2.2 Question Answering Sentence Selection

There exists substantial previous work on question answering sentence selection, or more generally, sentence pair modeling.

Syntactic and Semantic Analysis Bag of words representation with simple surface form matching often results in poor predictive power, leading to prior work exploring syntactic and semantic structures of the text. Bilotti et al. (2007) preprocessed the corpus with a semantic parser and an NER system. These semantic analyses are expressed as structural constraints on semantic annotations and keywords, and are translated directly into structured queries. Moldovan et al. (2007) transformed questions to logic representations based on their syntactic, semantic and contextual information, utilizing a logic prover to perform QA. Heilman and Smith (2010) and Yao, Van Durme, Callison-Burch, et al. (2013) used tree edit distance, and M. Wang, Smith, and Mitamura (2007) employed quasi-synchronous grammars to match the dependency parse trees of the question and the answer sentence.

Lexical semantic features Instead of utilizing higher-level abstractions such as syntactic and semantic analysis, another thread of previous work focussed on shallow lexical semantic features. Yih et al. (2013) performed semantic matching based on a latent word-alignment structure arising from WordNet. A. Lai and Hockenmaier (2014) utilized word relations such as words being synonyms, antonyms, hypernyms and hyponyms to perform a more fine-grained semantic overlap between sentences.

Neural methods for ranking L. Yu et al. (2014) and Severyn and Moschitti (2015) proposed the use of convolutional neural networks (CNNs) to model question and answer pairs, followed by Yang, Yih, and Meek (2015) with a related model and the introduction of the WikiQA dataset. Tan, Xiang, and Zhou (2015) and D. Wang and Nyberg (2015) made use of bidirectional LSTM networks to model question answer pairs. J. Rao, H. He, and J. J. Lin (2016) used either CNNs or RNNs to encode sentences, but utilized a triplet loss to learn to *rank* answer candidates. To better capture the interdependency between the question answer sentence pairs, Yin et al. (2016) proposed a generic attention-based CNN to model the sentence pairs for question answering, paraphrase identification and textual entailment. Amiri et al. (2016) presented a pairwise context-sensitive autoencoder to computing text pair similarity, and achieved state-of-the-art performance on answer reranking.

All of these efforts were aimed at candidate set re-ranking, once an initial retrieval step had been performed. P.-S. Huang et al. (2013) proposed to embed questions and the sentences of a provided corpus together into a shared vector space, followed by an "argmax" operation at query time to seek the sentence maximizing cosine similarity: they give no details on what is by default a linear operation in the size of the corpus, which is impractical for large collections as compared to our sub-linear retrieval approach.

5.3 General Approach

5.3.1 Problem Formulation

Formally, given a candidate set $C = \{p_1, \cdots, p_N\}$, a query q and a relevance function $F(q, p)$, an IR system retrieves the top-k items under the objective

$$\underset{p \in C}{\arg\operatorname{top}_k} F(q, p). \tag{5.4}$$

If the function F is simple enough (e.g. *tf-idf*), it could be easily solved by traditional IR techniques. However, tackling this problem with a complex F via straightforward application of supervised classification (e.g., recent neural network based models) requires a traversal over all possible candidates, i.e. the corpus, which is computationally infeasible for any reasonable collection.

5.3.2 Linear Featurized Model

To tackle the problem state above, we elaborate our design of a linear *featurized* model.

Let $\mathbf{f}_Q(q)$ refer to feature extraction on the query q, with corresponding candidate-side feature extraction $\mathbf{f}_P(p)$ on the candidate, and finally $\mathbf{f}_{QP}(q,p)$ extracts features from a (query, candidate) pair is defined in terms of $\mathbf{f}_Q$ and $\mathbf{f}_P$ via some form of composition (defined later):

$$\mathbf{f}_{QP}(q,p) = C(\mathbf{f}_Q(q), \mathbf{f}_P(p)). \tag{5.5}$$

From a set of query/candidate pairs we can train a model M such that given the feature vector of a pair (q,p), its returning value $f(\mathbf{f}_{QP}(q,p))$ represents the relevance score of whether the passage p answers the question q. In this work, the model is selected to be linear with the feature weight vector $\boldsymbol{\theta}$, leading to the optimization problem

$$\arg\max_{p\in\mathcal{D}} \boldsymbol{\theta} \cdot \mathbf{f}_{QP}(q,p). \tag{5.6}$$

This is in accordance with the pointwise reranker approach, and is an instance of the linear feature-based model of Metzler and Croft (2007).

Under specific compositional operations in $\mathbf{f}_{QP}$ the following transformation can be made:

$$\boldsymbol{\theta} \cdot \mathbf{f}_{QP}(q,p) = \mathbf{t}_{\boldsymbol{\theta}}(\mathbf{f}_Q(q)) \cdot \mathbf{f}_P(p). \tag{5.7}$$

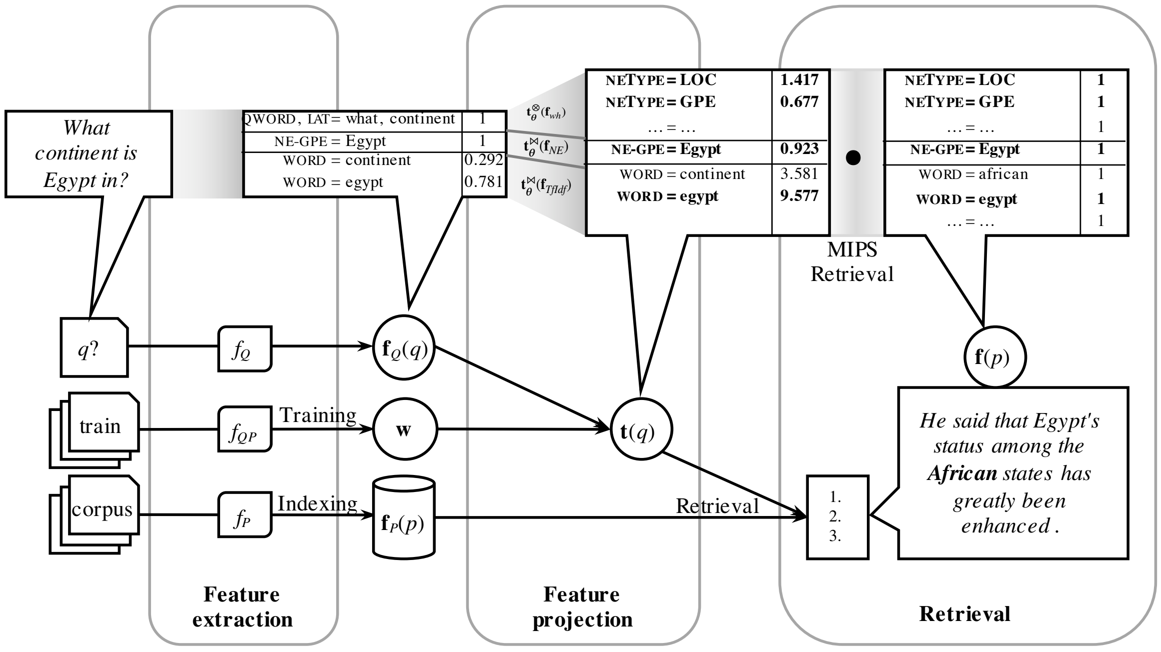


Figure 5.1: A schematic diagram of discriminative IR.

This is elaborated in § 4. We project the original feature vector of the query $\mathbf{f}_Q(q)$ to a transformed version $\mathbf{t}_{\boldsymbol{\theta}}(\mathbf{f}_Q(q))$: this transformed vector is dependent on the model parameters $\boldsymbol{\theta}$, where the association learned between the query and the candidate is incorporated into the transformed vector. This is a weighted, trainable generalization of *query expansion* in traditional IR systems.

Under this transformation we observe that the joint feature function $\mathbf{f}_{QP}(q, p)$ is decoupled into two parts with no interdependency – the original problem in Equation 5.7 is reduced to a standard **maximum inner product search** (MIPS) problem as seen on the RHS of Equation 5.7. Under sparse assumptions (where the query vector and the candidate feature vector are both sparse), this MIPS problem can be efficiently (sublinearly) solved using classical IR techniques (multiway merging of postings lists), where the index, instead of being a bag-of-words model, became a **bag-of-features** model instead.

A schematic diagram of this approach can be found in Figure 5.1.

5.4 Feature Composition Algebra

In this section we will devise a feature composition algebra that allows for such specific compositional operations. Some definitions will be introduced below.

5.4.1 Feature Vectors

A feature vector can be seen as an associative array (or in Python terminology, a `Dict`) that maps features in the form "KEY:*value*" to real-valued weights (each feature KEY:*value* will then be indexed to a unique ID so that the vector is a sparse vector). One item in a feature vector $\mathbf{f}$ is denoted as "$(\text{KEY} : \text{value}, \textit{weight})$", and a feature vector can be seen as a set of such tuples. We write $\mathbf{f}[\text{KEY} : \text{value}] = \textit{weight}$ to indicate that the features serve as keys to the associative array, and θ_X is the weight of the feature X in the trained model $\boldsymbol{\theta}$.

This feature definition subsumes a wide range of different featurization used in NLP. For example:

- Boolean feature with both key and value. For example, there could be a feature vector entry that says "there is a named entity with type GPE in this sentence." This would be represented as a feature $(\text{NAMED-ENTITY-TYPE} : \text{GPE}, 1)$;

- A Boolean feature that sigifies the distance between two words. The feature could take the form of $(\textsc{distance} : 5, 1)$;
- Weighted tf-idf bag-of-words features, e.g. $(\textsc{word} : \text{avocado}, 7.465)$;
- etc.

These show the expressiveness of this formulation of feature vectors. We will in turn develop our algebra based on this feature definition.

5.4.2 Feature Composition: Cartesian Product

For any feature vector $\mathbf{f} = \{(k : v, w)\}$ and $\mathbf{g} = \{(k' : v', w')\}$, the Cartesian product $\mathbf{f} \otimes \mathbf{g}$ of feature vectors $\mathbf{f}$ and $\mathbf{g}$ is defined as:

$$\mathbf{f} \otimes \mathbf{g} = \{((k, k') : (v, v'), ww'\} \, . \tag{5.8}$$

This Cartesian product can just be thought as the Cartesian product of two multisets. It can be used to measure the association between the query and the passage, under our scenario of question answering here (details see section below).

5.4.3 Feature Composition: Inner Join

Again, For any feature vector $\mathbf{f} = \{(k : v, w)\}$ and $\mathbf{g} = \{(k' : v', w')\}$, we define the *inner join* $\mathbf{f} \bowtie \mathbf{g}$ of feature vectors $\mathbf{f}$ and $\mathbf{g}$ as: [1]

$$\mathbf{f} \bowtie \mathbf{g} = \{(k = k' : \texttt{None}, ww') \mid v = v'\} . \tag{5.9}$$

5.4.4 Feature Projection for Cartesian Product

We define the projection of the feature vector $\mathbf{f}$ under the Cartesian product operation $\otimes$ and the linear model $\boldsymbol{\theta}$ as:

$$\mathbf{t}^{\otimes}_{\boldsymbol{\theta}}(\mathbf{f}) = \{(k' : v', w \cdot \theta[(k, k') : (v, v')]) \mid (k : v, w) \in \mathbf{f}\} , \tag{5.10}$$

for all k', v' such that $\theta[(k, k') : (v, v')] \neq 0$.

Theorem 5.1. *For any feature vector* $\mathbf{f}$ *and* $\mathbf{g}$*, we can decompose the score of the Cartesian product* $\boldsymbol{\theta} \cdot (\mathbf{f} \otimes \mathbf{g})$ *as*

$$\boldsymbol{\theta} \cdot (\mathbf{f} \otimes \mathbf{g}) = \mathbf{t}^{\otimes}_{\boldsymbol{\theta}}(\mathbf{f}) \cdot \mathbf{g} . \tag{5.11}$$

[1] The value of the join can only be `None`: This designates the *unit* type (or the 0-tuple) with only 1 possible value, variously denoted as `()`, `None`, or `Unit` in programming languages literature.

Proof.

$$\begin{aligned}
\boldsymbol{\theta} \cdot (\mathbf{f} \otimes \mathbf{g}) &= \boldsymbol{\theta} \cdot \{(k, k') : (v, v'), ww' \mid (k : v, w) \in \mathbf{f}, (k' : v', w') \in \mathbf{g}\} \\
&= \sum_{(k:v,w)\in\mathbf{f}} \sum_{(k':v',w')\in\mathbf{g}} ww' \cdot \theta[(k, k') : (v, v')] \\
&= \{(k' : v', w \cdot \theta[(k, k') : (v, v')] \mid (k : v, w) \in \mathbf{f}\} \cdot \{(k' : v', w') \mid (k' : v', w') \in \mathbf{g}\} \\
&= \mathbf{t}^{\otimes}_{\boldsymbol{\theta}}(\mathbf{f}) \cdot \mathbf{g} \,.
\end{aligned}$$

□

This shows that the score for the Cartesian product joint feature vector is indeed decomposable, as is shown in Equation 5.11.

5.4.5 Feature Projection for Inner Join

Similarly, we show that the score of the inner-joined feature vector can also be similarly decomposed.

We define the projection of the feature vector **f** under the inner join operation $\bowtie$ and the linear model $\boldsymbol{\theta}$ as:

$$\mathbf{t}^{\bowtie}_{\boldsymbol{\theta}}(\mathbf{f}) = \{(k' : v, w \cdot \theta[k = k' : \mathsf{None}]) \mid (k : v, w) \in \mathbf{f}\} \,, \tag{5.12}$$

for all k' such that $\theta[k = k' : \mathsf{None}] \neq 0$.

Theorem 5.2. *For any feature vector* $\mathbf{f}$ *and* $\mathbf{g}$*, we can decompose the score of the inner join* $\boldsymbol{\theta} \cdot (\mathbf{f} \bowtie \mathbf{g})$ *as*

$$\boldsymbol{\theta} \cdot (\mathbf{f} \bowtie \mathbf{g}) = \mathbf{t}_{\boldsymbol{\theta}}^{\bowtie}(\mathbf{f}) \cdot \mathbf{g} \,. \tag{5.13}$$

Proof.

$$\begin{aligned}
\boldsymbol{\theta} \cdot (\mathbf{f} \bowtie \mathbf{g}) &= \boldsymbol{\theta} \cdot \{(k' : v, w \cdot \theta[k = k' : \texttt{None}]) \mid (k : v, w) \in \mathbf{f}\} \\
&= \sum_{(k:v,w)\in\mathbf{f}} \sum_{\substack{(k':v',w')\in\mathbf{g} \\ v=v'}} ww' \cdot \theta[k = k' : \texttt{None}] \\
&= \{(k' : v, w \cdot \theta[k = k' : \texttt{None}]) \mid (k : v, w) \in \mathbf{f}\} \cdot \{(k' : v', w') \mid (k' : v', w') \in \mathbf{g}\} \\
&= \mathbf{t}_{\boldsymbol{\theta}}^{\bowtie}(\mathbf{f}) \cdot \mathbf{g} \,.
\end{aligned}$$

□

Again, this shows that the score of an inner joined feature vector can also be decomposed as an inner product of two sparse vectors, as is in Equation 5.13.

5.4.6 Feature Projection for Any Composed Features

Since both Cartesian product and inner product can be projected (by Theorem 5.1 and Theorem 5.2), we show that any composed feature vector that is the sum of these two kinds of composition can be projected.

Theorem 5.3. *For any feature vector* $\mathbf{f}$ *with a number of components* $\mathbf{f} = \sum_{i=1}^{m} \mathbf{f}_i$ *and feature* *vector* $\mathbf{g}$*, also with a number of components* $\mathbf{g} = \sum_{j=1}^{n} \mathbf{g}_j$*, a composition feature function* $C(\mathbf{f}, \mathbf{g})$ *that is the sum of various Cartesian or inner product composition of any components between* $\mathbf{f}$ *and* $\mathbf{g}$*, namely* $C(\mathbf{f}, \mathbf{g}) = \sum_{k=1}^{K} \mathbf{f}_{l_k} \diamond_k \mathbf{g}_{r_k}$*, where* $1 \leq l_k \leq m$ *and* $1 \leq r_k \leq n$ *are the indices to the components in* $\mathbf{f}$ *and* $\mathbf{g}$*, and* $\diamond_k \in \{\otimes, \bowtie\}$ *are any of the composition operators, can be projected. That is, there exist a projection function* $\mathbf{t}_{\boldsymbol{\theta}}^{C}(\cdot)$ *such that*

$$\boldsymbol{\theta} \cdot C(\mathbf{f}, \mathbf{g}) = \mathbf{t}_{\boldsymbol{\theta}}^{C}(\mathbf{f}) \cdot \mathbf{g} \,. \tag{5.14}$$

Proof. Construct a projection function

$$\mathbf{t}_{\boldsymbol{\theta}}^{C}(\mathbf{f}) = \sum_{k=1}^{K} \mathbf{t}_{\boldsymbol{\theta}}^{\diamond_k}(\mathbf{f}_{l_k}) \,. \tag{5.15}$$

We have

$$\begin{aligned}
\boldsymbol{\theta} \cdot C(\mathbf{f}, \mathbf{g}) &= \sum_{k=1}^{K} \boldsymbol{\theta} \cdot (\mathbf{f}_{l_k} \diamond_k \mathbf{g}_{r_k}) \\
&= \sum_{k=1}^{K} \mathbf{t}_{\boldsymbol{\theta}}^{\diamond_k}(\mathbf{f}_{l_k}) \cdot \mathbf{g}_{r_k} \\
&= \sum_{k=1}^{K} \mathbf{t}_{\boldsymbol{\theta}}^{\diamond_k}(\mathbf{f}_{l_k}) \cdot \mathbf{g} \\
&= \mathbf{t}_{\boldsymbol{\theta}}^{C}(\mathbf{f}) \cdot \mathbf{g} \,.
\end{aligned}$$

Hence we have $\boldsymbol{\theta} \cdot C(\mathbf{f}, \mathbf{g}) = \mathbf{t}_{\boldsymbol{\theta}}^{C}(\mathbf{f}) \cdot \mathbf{g}$ (Equation 5.14) as desired.

□

5.5 Features for Question Answering

We first describe the feature functions (featurizers: a function, given an object, returns a feature vector) used in QA retrieval here to lay the foundation for further discussion to be grounded.

For natural language questions, we extract the following types of features:

Question word ($\mathbf{f_{wh}}$) The type of the question, typically the *wh*-word of the question sentence. If it is a question that starts with something like "*How many*", the word after the question word *how* is also included in the feature. It results in a Boolean feature, e.g. $(\text{Q-WORD} : \text{how many}, 1)$.

Lexical answer type ($\mathbf{f_{lat}}$) If the query is a question where the question word is either "what" or "which", the *lexical answer type* (LAT) is question is identified (Ferrucci et al., 2010), which is defined as the head word of the first NP (noun phrase) after the question word. For example, the LAT feature from the question "*What is the city of brotherly love?*" would be $(\text{LAT} : \text{city}, 1)$.

Typed named entities ($\mathbf{f_{NE}}$) All the named entities discovered in the question are extracted (its type also form part of the feature). For example, if a named entity "*Margaret Thatcher*" is detected with named entity type PERSON, a feature $(\text{NE-PERSON} : \text{Margaret Thatcher}, 1)$ will be generated.

TF-IDF-weighted bag-of-words for queries ($\mathbf{f}_{\textbf{TF-IDF}}$) The TF-IDF-weighted bag-of-words feature for a natural language question. An example feature would be (WORD : author, 0.454), denoting that the word "*author*" has a weight of 0.454 in the sentence. Note that this feature is real-valued.

Similarly, for candidate passages, we can also define the following types of feature extractors:

Bag-of-words features ($\mathbf{f}_{\textbf{BoW}}$) Bag-of-words: any distinct word x in the passage will generate a feature $(\text{WORD} : x, 1)$.

Named entity types ($\mathbf{f}_{\textbf{NE-Type}}$) Named entity type. If the passage contains a name of a person, a feature ($\text{NE-TYPE} : \text{PERSON}$, 1) will be generated.

Given these features, we could define the following joint feature extractor for question answering reranking:

$$
\begin{array}{llcll}
C(& \mathbf{f}_Q(q) & , & \mathbf{f}_P(p) &) \\
= & \mathbf{f}_{\text{wh}}(q) & \otimes & \mathbf{f}_{\text{NE-Type}}(p) & \\
+ & (\mathbf{f}_{\text{wh}}(q) \otimes \mathbf{f}_{lat}(q)) & \otimes & \mathbf{f}_{\text{NE-Type}}(p) & \\
+ & (\mathbf{f}_{\text{wh}}(q) \otimes \mathbf{f}_{\text{lat}}(q)) & \otimes & \mathbf{f}_{\text{BoW}}(p) & \\
+ & \mathbf{f}_{\text{NE}}(q) & \bowtie & \mathbf{f}_{\text{NE}}(p) & \\
+ & \mathbf{f}_{\text{TF-IDF}}(q) & \bowtie & \mathbf{f}_{\text{BoW}}(p) & .
\end{array}
\tag{5.16}
$$

We will dissect these feature compositions in detail below.

$\mathbf{f}_{\text{wh}}(q) \otimes \mathbf{f}_{\text{NE-Type}}(p)$ captures the association of question words and the expected type of named entities. During training, we discovered features like $(\textsc{q-word}, \textsc{ne-type}) :$ (who, $\texttt{PERSON}$), or $(\textsc{q-word}, \textsc{ne-type}) :$ (when, $\texttt{DATE}$) will be assigned high weights. This shows that the model learns, for example, to answer a question that asks "*when*", an answer sentence with a DATE should be of higher relevance.

$(\mathbf{f}_{\text{wh}}(q) \otimes \mathbf{f}_{\text{lat}}(q)) \otimes \mathbf{f}_{\text{NE-Type}}(p)$ captures the association of question words *together with* lexical answer types with the expected type of named entities, in case that the question word alone is not enough. This would result in the following example high-weighted features to be learned: $((\textsc{q-word}, \textsc{lat}), \textsc{ne-type}) : ((\text{what}, \text{city}), \texttt{GPE})$.

$(\mathbf{f}_{\text{wh}}(q) \otimes \mathbf{f}_{\text{lat}}(q)) \otimes \mathbf{f}_{\text{BoW}}(p)$ captures the relation between some question types with certain words in the answer. For example, we observed feature "$((\textsc{q-word}, \textsc{lat}), \textsc{word}) : ((\text{what}, \text{capacity}), \text{gallon})$" to have a relative high weight, because the word "gallon" can be expected from a question asking about capacity.

$\mathbf{f}_{\text{NE}}(q) \bowtie \mathbf{f}_{\text{NE}}(p)$ captures named entity overlap. Features like $(\textsc{ne-person} = \textsc{ne-person}) :$ $\texttt{None}$ will be assigned high weights because sentences talking about the same person will have high question-answer association. Interestingly, we observed feature $(\textsc{ne-norp}^2 = \textsc{ne-language}) = 1$ is of high weight, because words like "French" can refer to either a language or an adjective meaning "pertaining to France". This kind of feature helps mitigates the error of named entity annotations.[3]

[2]NORP: Nationality.

[3]As observed by Yao et al.

$\mathbf{f}_{\text{TF-IDF}}(q) \bowtie \mathbf{f}_{\text{BoW}}(p)$ measures general *tf-idf*-weighted context word overlap. Using only this feature without the others effectively reduces the system to a traditional *tf-idf*-based retrieval system.

5.6 Training

The model is trained to maximize the likelihood of correct answer sentences answering the given questions:

$$P(p \mid q) = \frac{1}{1 + \exp(-F(q, p))} . \tag{5.17}$$

where the relevance function $F(q, p) = \boldsymbol{\theta} \cdot C(\mathbf{f}_Q(q), \mathbf{f}_P(p)) = \mathbf{t}^C_{\boldsymbol{\theta}}(\mathbf{f}_Q(q)) \cdot \mathbf{f}_P(p)$.

We minimize the corresponding negative log likelihood with L_1 (LASSO) regularization to enforce *sparsity* on the model so that only important features are nonzero: this ensures that the projected feature vector $\mathbf{t}^C_{\boldsymbol{\theta}}(\mathbf{f}_Q(q))$ is also sparse. After all, the reduction to bag-of-feature retrieval requires the query vector to be sparse to ensure good performance.

The L_1 regularization coefficient λ is a hyperparameter that controls the sparseness of the model: It will be tuned on the dev set.

5.7 Retrieval

We use Apache LUCENE[4] to build the index of the corpus, which, in the scenario of this work, is the feature vectors of all candidates $\mathbf{f}_P(p), p \in \mathcal{D}$. This is an instance of weighted bag-of-features instead of common bag-of-words.

For a given question q, we first compute its feature vector $\mathbf{f}(q)$ and then compute its transformed feature vector $\mathbf{t}_{\boldsymbol{\theta}}(q)$ given model parameters $\boldsymbol{\theta}$, forming a weighted query. We modified the similarity function of LUCENE when executing multiway postings list merging so that fast efficient maximum inner product search can be achieved. This classical IR technique ensures sublinear performance because only vectors with at least one overlapping feature, instead of the whole corpus, is traversed.[5] For details, see Section A.1.

5.8 Experiments

5.8.1 Datasets

TREC/AQUAINT Data We use the training and test data from Yao, Van Durme, and P. Clark (2013). Passages are retrieved from the AQUAINT Corpus (Graff, 2002), which is NER-tagged by the Illinois Named Entity Tagger (Ratinov and Roth, 2009) with an 18-label entity type set. Questions are parsed using the Stanford CORENLP (Manning et

[4] `http://lucene.apache.org`.

[5] The closest work on indexing we are aware of is by Bilotti et al. (2007), who transformed linguistic structures to structured constraints, which is different from our approach of directly indexing linguistic features.

Dataset	# of questions train	dev	test	# of sentences
TREC/AQUAINT	2,150	53	99	23,398,942
WIKIQA/Wikipedia	2,118	77	157	20,368,761

Table 5.1: Summary of the datasets.

al., 2014) package. Each question is paired with 10 answer candidates from AQUAINT, annotated for whether it answers the question via crowdsourcing. The test data derives from J. J. Lin and Katz (2006), which contains 99 TREC questions that can be answered in AQUAINT. We follow Nallapati (2004) and undersample the negative class, taking 50 sentences uniformly at random from the AQUAINT corpus, per query, filtered to ensure no such sentence matches a query's answer pattern as negative samples to the training set. The summary of the datasets are shown in Table 5.1.[6]

WikiQA-Wikipedia Data We introduce a novel evaluation dataset for QA retrieval, based on WIKIQA (Yang, Yih, and Meek, 2015), which pairs questions asked to Bing with their most associated Wikipedia article, along with sentence-level annotations on the introductory section of those articles as to whether they answer the question.

Note that as compared to the TREC dataset, there are some questions in WIKIQA which are not answerable based on the provided context alone: rather, they require outside commonsense knowledge to answer. For example, the question "*Who is the guy in the wheelchair who is smart*" has the answer snippet "Professor Stephen Hawking, known for

[6]The number of negative samples of the training set does not include the randomly sampled 50 negative samples for each training question.

being a theoretical physicist, has appeared in many works of popular culture." This sets the upper bound on performance with WIKIQA below 100 when using contemporary question answering techniques, as assumed here.

We automatically aligned WIKIQA annotations, which was based on an unreported version of Wikipedia, with the Feb. 2016 snapshot, using for our corpus the introductory section of *all* Wikipedia articles, processed with Stanford CORENLP. Alignment was performed via string edit distance, leading to a 55 alignment to the original annotations. Table 1 dev/test reflects the subset resulting from this alignment; all of the original WIKIQA train was used in training, along with 50 negative examples randomly sampled per question.

5.8.2 Setup and Baseline Systems

The model is trained using LIBLINEAR (Fan et al., 2008) with heavy L_1-regularization (feature selection) to the maximum likelihood objective. The model is tuned on the dev set, with the objective of maximizing recall.

Recent work in neural network based *reranking* is not directly applicable here as those are *linear* with respect to the number of candidate sentences, which is computationally infeasible given a large corpus. The following baseline systems are compared against:

- **Off-the-shelf LUCENE**: Directly indexing the sentences in LUCENE and perform sentence retrieval. This is equivalent to maximum *tf-idf* retrieval.

- **Yao, Van Durme, and P. Clark (2013)**: A retrieval system which augments the bag-of-words query with desired named entity types based on a given question. This can be seen as a special case of the method described in this chapter.

5.8.3 Metrics

We use the following metrics to evaluate the performance of various retrieval systems for question answering.

- **R@1k**: The recall in top-1000 retrieved list. Contrary to normal IR systems which optimize precision (as seen in metrics such as P@10), our system is a triaging system whose goal is to *retrieve good candidates* for downstream reranking: high recall within a large set of initial candidates is our foremost aim.
- **b-pref** (Buckley and Voorhees, 2004): This is designed for situations where relevance judgments are known to be far from complete, computing a preference relation of whether judged relevant documents are retrieved ahead of judged irrelevant document. This is usually the case in passage retrieval, where complete annotation of all sentences in a large corpus as to whether they answer each question is not feasible beyond a small set (such as the work of J. J. Lin and Katz (2006), as we have utilized here in the TREC/AQUAINT corpus).
- **MAP**: mean average precision;
- **MRR**: mean reciprocal rank.

Model	**R@1k**	**b-pref**	MAP	MRR
TREC / AQUAINT				
LUCENE (dev)	52.44	41.95	9.63	13.94
LUCENE (test)	35.47	38.22	9.78	15.06
Yao+ (test)	25.88	45.41	13.75	**29.87**
DiscIR (dev)	71.34	70.69	20.07	30.34
DiscIR (test)	**78.20**	**75.15**	**17.84**	25.30
WIKIQA / Wikipedia				
LUCENE (dev)	25.00	25.97	1.83	1.83
LUCENE (test)	24.73	25.69	0.58	0.72
DiscIR (dev)	60.00	61.69	9.56	9.65
DiscIR (test)	**58.79**	**60.88**	**10.26**	**11.42**

Table 5.2: Performance of the QA retrieval systems.

We are most concerned with R@1k and b-pref here since our focus is recall. MAP and MRR are reported in keeping with prior work.

5.8.4 Results

Our approach (DiscIR) significantly outperforms Yao, Van Durme, and P. Clark (2013) in R@1k and b-pref, demonstrating the effectiveness of trained weighted queries compared to binary augmented features. The performance gain with respect to off-the-shelf LUCENE with reranking shows that our weighted augmented queries by decomposition is superior to vanilla *tf-idf* retrieval, as can be shown in Table 5.2.

We also plot the performance of these systems at different ks on a log-scale (see Figure 5.2 and Figure 5.3). We use two metrics here: recall at k (R@k) and success at k (S@k). Success at k is the percentage of queries in which there was at least one relevant answer sentence among the first *k* retrieved result by a specific system, which is the true upper bound for downstream tasks.

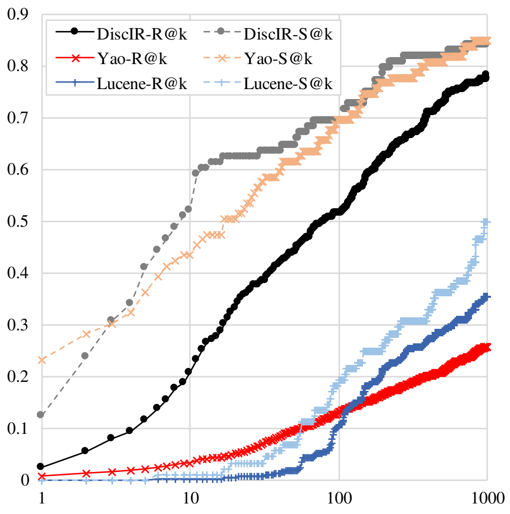


Figure 5.2: The R@*k* and S@*k* curve for different models in the TREC/AQUAINT setting.

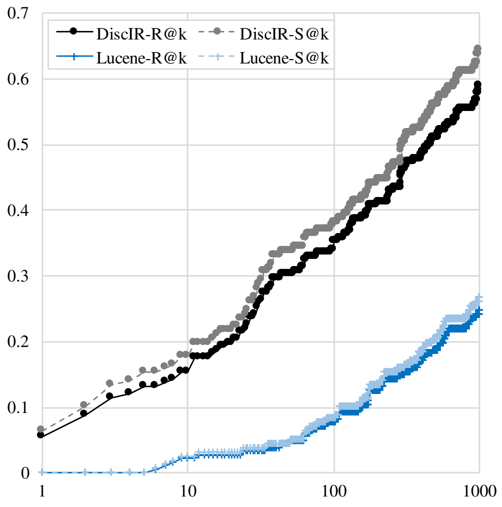


Figure 5.3: The R@*k* and S@*k* curve for different models in the WIKIQA/Wikipedia setting.

Again, DiscIR demonstrated significantly higher recalls than baselines at different *k*s and across different datasets. Success rate at different *k*'s are also uniformly higher than LUCENE, and at most *k*'s higher than the model of Yao et al.'s.

5.8.5 End-to-end Performance after Reranking

As we stated before in this thesis, a first-stage retriever's job is to find as many relevant candidates as possible so that downstream rerankers can rank them high on the ranked list. In this section we investigate the effect of a state-of-the-art reranker under the retrieved result of BM25 and our Discriminative IR models, using the TREC/AQUAINT data.

We train a BERT-based reranker $F_{\mathrm{BERT}}(q,c)$ that maps a natural language question and an answer candidate to a score indicating whether c answers the question in q. The model concatenates the query and the candidate answer sentence with delimiters, and pass the sequence through BERT. The score is derived from the first CLS token (similar to the UNLI model we proposed earlier in Chapter 3). We train this via pairwise ranking, dictating the model to rank relevant candidates above irrelevant candidates for answer sentence selection (J. Rao, H. He, and J. J. Lin, 2016), using the TREC-QA dataset. The model achieves 89.7% MAP on the TREC-QA test set, similar to recent models with BERT-style pretraining models (T. M. Lai et al., 2019).

We execute the pipeline of first retrieving top-K using either BM25 or our DiscIR here as the relevance function, and then rerank the retrieved list using the BERT-based reranker, tested on the dev set of TREC/AQUAINT IR corpus. For $K = 1024$, reranking raised MAP from 20.1% to 70.0% for DiscIR, and from 13.8% to 51.7% for vanilla BM25 (for other K's, see Figure 5.4). This shows that the improved recall of DiscIR has a significant improvement over traditional vanilla IR, boosting downstream reranking performance.

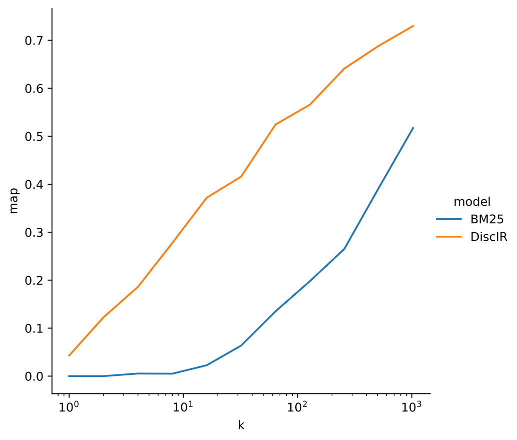


Figure 5.4: Final MAP score after BERT-based reranking, with different K (log-scale) for triaging.

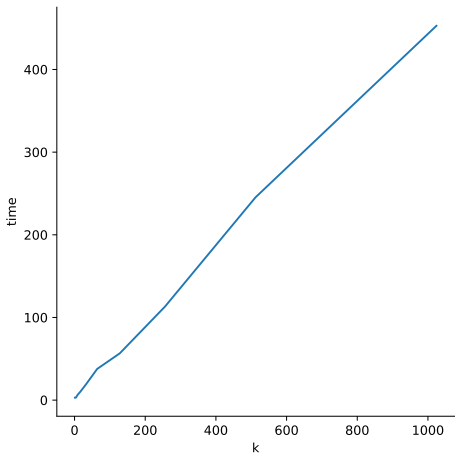


Figure 5.5: Average time consumed for executing each query.

There exist the tradeoff between performance and latency: the larger K is, the higher recall the triaging phase shows, the slower the pipeline runs since all the K retrieved candidates are going to be reranked using the heavyweight neural reranker. We plot the average processing time for each query, with $K \in \{1, 2, 4, \cdots, 1024\}$ in Figure 5.5. We could see that the processing time is mostly linear with respect to K; and at $K = 1024$, the average processing time for a query is 453ms on a machine with one NVidia GTX 1080Ti GPU. The selection for K depends on hardware constraints: for example, at $K = 128$, the average processing time is 56ms, but the MAP is already 56.6% for DiscIR, significantly higher than BM25 at the same K, whose MAP is only 19.8%. The selection of K depends on such tradeoff, and will be determined by real-world engineering constraints.

5.8.6 Error Analysis

We showcase some typical negative samples of our discriminative IR system running over the question answering sentence retrieval tasks. Most of these errors arise from the lack of expressiveness of the feature functions, as can be shown below.

Q: What is the abbreviation of London Stock Exchange?

A: The London Stock Exchange (LSE) board Thursday agreed to introduce its controversial computerized order-driven share dealing system, but not for at least another year and initially only for FT-SE 100 stocks.

Failed because the current feature set is unable to capture how to answer an "abbreviation" question. This is a case of lacking understanding of the word "*abbreviation*" and the parenthesis construction. If we add an additional lightweight feature to detect all-caps words in the feature side ($\mathbf{f}_{\text{allCaps}}$ will fire if there exists an all-caps word in the candidate sentence), and add a feature $(\mathbf{f}_{\text{wh}} \otimes \mathbf{f}_{\text{lat}}) \otimes \mathbf{f}_{\text{allCaps}}$, given enough training data, a feature ((Q-WORD, LAT), ALL-CAPS) : ((what, abbreviation), TRUE) will probably gain a high weight in the training process, hence enabling the system to correctly answer this question instance.

Q: What is another name for the North Star?

A: Capella is important to navigators here on Earth, as it is the closest bright star to the North Star, Polaris in Ursa Minor, and it can be seen for at least part of the night every month.

Failed because the model in unable to capture how to answer a question asking for appositives. To solve this problem, features that captures the relation of appositives must be present. These semantic features are heavyweight features that requires heavy preprocessing on the passage side, hence would result in big overhead in the indexing of the corpus.

Q: What is the fastest car in the world?

A: The <u>Thrust SuperSonic Car</u> set a world land speed record of 1,142 kph on September 25.

Our approach failed to retrieve this sentence because of the inability of the feature set to learn the association between "fastest" and "speed record." This is best solved by distributed semantic representation of words and sentences, which are inherently dense, real-valued vectors. Because this work relies on sparsity to achieve its sublinear retrieving performance, these dense features are out of scope for this sparse approach to tackle.

These errors can be classified into several bins:

- Lacking lexical understanding: Two words might be synonymous, but our lexical features cannot represent them (e.g. the query uses the word "created" but the relevant answer has the word "established"). These might be alleviated by term-level query expansion;

- Superficial matching: Our retriever might match *Mount Victoria* with *capital of Victoria*, in which *Victoria* is clearly a geo-political entity instead of a mountain. These errors are instances of superficial matching while lacking understanding of context;

- Lacking syntactic understanding: The abbreviation and the appositive example above are examples of this kind of errors;
- Lacking semantic understanding: The “fastest car” example above is an example of this class of error: our feature functions cannot understand the semantic relevance of “the fastest car” and “setting a world land record”. These might be ameliorated by using neural representations elaborated in the next chapter.

We manually sample and inspect 50 queries in both TREC-QA and the WikiQA datasets whose first relevant answer is not retrieved in their corresponding top 10 retrieved candidate set (in other words, for these queries, their P@10 metric equals 0), and classifies these errors into the types of errors described above:

Error type	**Proportion**
Lacking lexical understanding	18%
Superficial matching	30%
Lacking syntactic understanding	8%
Lacking semantic understanding	44%

Table 5.3: Proportions of common errors made by our discrimative QA retrieval system.

5.9 Conclusion

Yao, Van Durme, and P. Clark (2013) proposed to couple information retrieval with features from downstream question answer sentence selection. We generalized this intuition by recognizing it as an instance of discriminative retrieval, and proposed a new framework for generating weighted, feature-rich queries based on a given query (may be a natural language question or a mention as we discussed in this thesis). This approach allows for the straightforward use of a downstream model in the candidate selection process, and leads to a significant gain in recall, b-pref and MAP in the triaging step compared to prior work, hence providing better candidates for downstream reranking models, which could be coupled to this approach in future work.

Our framework is general and should apply to a variety of other structurally related tasks, such as entity linking (retrieving candidate entities from a large knowledge base given a query mention) and slot filling (retrieving candidate mentions from a large text corpus given a query mention and a relation).

This chapter concludes our discussion with sparse vector-based retrieval for question answering. In the next chapter, we extend this thread of research to dense vectors by harnessing recent advancements in neural representation learning.

Chapter 6

Retrieval with Dense Vectors

The previous chapter developed a sparse-vector based trainable retrieval system. However, some distributional semantics cannot be easily captured by sparse features, as the error analysis section has shown. In this chapter we turn to pure neural methods that learn dense vector representations for text and reuses these vectors for retrieval.

We explore using trainable dense vector-based neural representations for queries and candidates, and illustrate how such an efficient retrieval system can be built. While our exploration here does not yield a positive result, later research down this track has shown this is feasible and can outperform lexical retrieval under better strategies of negative sampling. Differences of these work with this chapter will be discussed.

6.1 Introduction

As we have discussed in previous chapters in detail, triaging systems commonly use bag-of-word or bag-of-feature (T. Chen and Van Durme, 2017) relevance functions that computes the relevance between a query and a candidate with heuristics defined over *lexical* or *feature* overlap. Though commonly employed models such as Okapi BM25 (Robertson, Walker, et al., 1994) are successful, these sparse vector-based retrieval systems still struggle to understand queries and documents beyond surface lexical or feature forms. A main issue is *vocabulary mismatch*, where the same concept is described with different words in the query and the candidate document. Under this scenario, lexical or feature overlap will not capture this kind of semantic relevance.

Furthermore, triaging systems for question answering requires the relevance function abilities to express semantic relatedness beyond what *ad hoc* retrieval demands: as we have shown in the previous chapter, using lexical retrieval systems like BM25 hinders the recall of the retriever, thus hurting the overall performance of the question answering system.

This motivates the development of neural representations for text, together with dense vector-based retrieval methods. We apply this method to two tasks:

- **Question answering sentence retrieval:** The same task in Chapter 5, but with dense encodings of questions and answer candidates learned;
- **Similar question retrieval:** Given a natural language question, retrieve similar questions in the corpus. This has real-world use: consider a social question answering

website like Quora—if a user asks a new question, the web interface can return similar questions and ask the user if the question has already been asked by other users. This reduces the question redundancy on these websites.

As we have discussed in Chapter 2, such a dense vector based retrieval system has to be equipped with *decoupled* encoders for both the query and the candidate: A $\mathbf{f}_Q : \mathcal{Q} \rightarrow \mathcal{R}_Q$ and a separate $\mathbf{f}_C : \mathcal{C} \rightarrow \mathcal{R}_C$. The two tasks here represents two different strategies with respect to how model parameters are tied. In the question answering case, the questions and answers are clearly different and should use two distinct set of parameters to model, whereas in the similar question retrieval case, question q_1 being similar to question q_2 implies that q_2 is also similar to q_1. We term the former case as *asymmetric* retrieval and the latter case as *symmetric* retrieval.

6.2 Method

Our dense vector based relevance function for a query q and a candidate c is

$$F(q,c) = \mathbf{q}^{\mathrm{T}}\mathbf{c}\ . \tag{6.1}$$

Here $\mathbf{q} \in \mathbb{R}^d$ is a fixed-length dense vector representation of q, and $\mathbf{c} \in \mathbb{R}^d$ is a fixed-length dense vector representation of c. Under this formalism, the retrieval problem is reduced to a maximum inner product search (MIPS) problem if the representations of candidates $\{\mathbf{c} \mid c \in C\}$ are stored and indexed.

6.2.1 Sentence Representations

We adopt the ELMo encoder (Peters et al., 2018) as our text contextualizer to take advantage of recent language model-based pretraining. For each sentence $s = (x_1, \cdots, x_n)$ (x_i's are tokens), we pass through the 2-layer bi-directional LSTM layers and take the max pooling to get a fixed-length vector. Under ELMo, the dimensionality $d = 1024$.

$$\mathbf{s} = \mathbf{f}(s) = \text{MaxPool}(\text{ELMo}(x_1, \cdots, x_n)) \in \mathbb{R}^d \; . \tag{6.2}$$

For the asymmetric and symmetric cases discussed above, we take different approaches here:

- **Asymmetric case:** Two copies of the encoder, one for queries and one for candidates, are both initialized with pre-trained ELMo parameters. At training time the two encoders are **not tied** so that different parameters can be learned for queries and candidates (in the task of question answering here, questions are queries and answer snippets are candidates):

$$F(q, c) = \mathbf{f}_Q(q) \cdot \mathbf{f}_C(c) \; ; \tag{6.3}$$

- **Symmetric case:** A single copy of the encoder for both queries and candidates is initialized with the pre-trained ELMo parameters. Neural parameters for questions and answers are tied: this makes sense since in our similar question retrieval scenario, queries and candidates are both natural language questions, and the relevance function is symmetric:

$$F(q,c) = \mathbf{f}_Q(q) \cdot \mathbf{f}_Q(c)\ . \tag{6.4}$$

6.2.2 Training

We use the pairwise ranking method (see Chapter 2) to train the models. A concern in these methods is *negative sampling*, or how to construct the training set of preference pairs.

Assume that each query q in the dataset is associated with a positive candidate set $C(q)$ where every $c \in C$ is considered as a relevant candidate. We construct two different negative sample sets for each query:

- **Random negative samples:** We uniformly sample k_1 candidates in the whole corpus set that is not positive (i.e., in C) as the uniformly negative sample set $C'_1(q)$;

- **Competitive negative samples:** The C'_1 negative set may be too easy for the model to learn since the sentences may differ a lot from the correct answers just be looking at its words. We mine some harder negative samples for the model by retrieving the top-k_2 using an off-the-shelf IR engine under a lexical overlap relevance function (BM25) from the corpus. Positive samples are removed accordingly. This set is negative (irrelevant to the query) but has considerable word-level overlap with the query, making it a set of harder negative samples for the model:

$$C'_2(q) = \underset{c' \in \mathcal{C} \setminus C}{\arg\operatorname{top}_{k_2}} \operatorname{BM25}(q, c') \tag{6.5}$$

The whole training set containing all the preference pairs would then be

$$\mathcal{R} = \bigcup_q \{(c, c') \mid c \in C(q), c' \in C'_1(q) \cup C'_2(q)\} . \tag{6.6}$$

Under the pairwise training loss, we minimize

$$L = \sum_{(c,c') \in \mathcal{R}} \max(0, \xi - F(q, c) + F(q, c')) + \frac{\lambda}{2} \|\mathbf{\Theta}\|_2^2 . \tag{6.7}$$

Note that we use L_2 regularization instead of L_1 used in Chapter 5 since sparsity is not desired in our dense retrieval scenario. Hyperparameters to tune on the development set include the desired margin ξ, the regularization coefficient λ, the number of uniformly sampled negative samples for each query k_1, and the number of negative samples that are competitive under BM25 k_2.

6.2.3 Retrieval

We use FAISS (Johnson, Douze, and Jégou, 2017) as our retrieval engine, with optimized product quantization (Ge et al., 2013) for maximum inner product search, where

100,000 samples are used to train the index (as Ge et al. (2013) required, a subset of the candidate dense vectors is used to train a PCA which is in turn used to normalize the space of the candidate vectors). The system runs efficiently under a single CPU.

6.3 Experiments on Question Answering

Datasets We here reuse the TREC/AQUAINT dataset created in the past chapter. The details can be found in Table 5.1.

Baselines and Setup We use BM25 as is implemented in Apache Lucene, Yao, Van Durme, and P. Clark (2013), and the Discriminative IR result elaborated in the previous chapter as baselines. We apply dropout probability $p_{\mathrm{D}} = 0.3$ for the query and candidate vectors, number of uniformly-sampled negative samples $k_1 = 40$, and margin hyperparameter $\xi = 0.5$.[1] We do not augment the preference pair set with Lucene-retrieved competitive negative samples, since in the dataset there are already competitive negative samples (for each query there are on average 6 of them).

Results Results are shown in Table 6.1. Clearly, our trained dense retriever did not outperform the lexical retrieval baseline BM25 except MRR, and it is easily outperformed by

[1] One might argue that the ξ can just be set as 1 in a normal linear SVM formulation. However, since our embedding results from a max pooling over ELMo outputs (which are LSTM outputs), its range is fixed in $[-1, 1]$ due to the tanh layer, it is not free as in a normal SVM, where the last layer is a linear layer with output range $\mathbb{R}$.

Model	R@1k	b-pref	MAP	MRR
BM25	35.5	38.2	9.8	15.1
Yao, Van Durme, and P. Clark (2013)	25.9	45.4	13.8	**29.9**
Discriminative IR	**78.2**	**75.2**	**17.8**	25.3
Dense Retrieval	19.3	19.3	6.5	15.9

Table 6.1: Results of retrieval methods over the TREC/AQUAINT dataset.

feature-augmented methods like Yao, Van Durme, and P. Clark (2013) and Discriminative IR (T. Chen and Van Durme, 2017) in the previous chapter.

6.4 Experiments on Similar Question Retrieval

Dataset Creation We use the Quora Question Pairs (QQP) dataset (Iyer, Dandekar, and Csernai, 2017). This dataset comprises of pairs of actual questions on Twitter, with a binary true/false annotation that indicates whether the two questions in the pair are similar or not.

We convert this dataset to a retrieval dataset. QQP data samples take the form $(q_1, q_2, y) \in D$ where $q_1, q_2 \in Q$ are natural language questions, and $y \in \{0, 1\}$ is the label. We say that q_1 and q_2 are similar, $q_1 \sim q_2$, if $(q_1, q_2, 1) \in D$.

Assuming that the question similarity relation is transitive, we construct an equivalence relation over Q by applying the transitive closure: if there exists a q_2 such that $q_1 \sim q_2$ and $q_2 \sim q_3$, we believe that $q_1 \sim q_3$. Under this equivalence relation "$\sim$", each question q gets its equivalence class $[q] \in Q/{\sim}$, the set of all questions similar to q. For a retrieval

Original QQP training similar pairs	363,871
Original QQP dev similar pairs	40,432
Original QQP test similar pairs	390,965
QQP distinct questions (also the candidate question set)	537,933
Similar candidate questions per query question (except itself)	1.20
QQP-Retrieval training queries	48,273
QQP-Retrieval dev queries	1,816
QQP-Retrieval test queries	1,816

Table 6.2: Statistics of the QQP and the transformed QQP-Retrieval datasets.

system, all questions in $[q]$ are considered as relevant, and thus should rank higher than all other irrelvant systems.

We call this retrieval dataset as QQP-Retrieval, and its statistics is shown in Table 6.2.

In QQP-Retrieval, each query q itself is in the question candidate set: $q \in C$, and itself is the most similar candidate. Except itself, each query has 1.20 similar questions in the candidate set on average. The task is to retrieve these similar question candidates. When evaluating, for each query q, the top candidate q itself is removed: since it is trivial, any downstream metrics like MAP and MRR will not take q into account.

Baselines and Setup Again, we use BM25 as is implemented in Apache Lucene as the off-the-shelf baseline. When training the model, we apply dropout rate for the encoder $p_\text{D} = 0.3$, number of uniformly-sampled negative samples $k_1 = 40$, number of Lucene-retrieved competitive negative samples $k_2 = 10$, and margin hyperparameter $\xi = 0.7$.

Model	MAP	MRR	R@5	R@1k
Lucene	**77.8**	**78.0**	**85.9**	**97.4**
Dense Retrieval	44.8	66.3	51.0	77.9

Table 6.3: Results of retrieval methods over the QQP-Retrieval dataset.

Results Results are shown in Table 6.3. Clearly, our trained dense retriever did not outperform the lexical retrieval baseline BM25.

6.5 Discussions

Our models were built in 2018 before any known publication that addressed this problem. Although it failed to outperform simple baselines such as BM25 in the question answering system, similar approaches have been shown to be successful (outperforming BM25) after this work, starting with the publication of Lee, Chang, and Toutanova (2019).

There are many similarities between the approach in this thesis chapter and the one in Lee, Chang, and Toutanova (2019), yet there are important differences. We hypothesize that Lee, Chang, and Toutanova (2019)'s success is based on a few factors different from our work here:

- The use of BERT (Devlin et al., 2019) for sentence encoding, leading to better dense vector representations of questions and answer candidates, as opposed to our ELMo-based vectors;
- Massive pretraining via their ICT strategy: They mine lots of *proxy* question and

answer pairs by sampling one sentence from a Wikipedia paragraph as the question, and all the other sentences as a proxy answer candidate. This strategy forces the model to learn the relevance function of question answerability instead of focusing on surface-level word overlap. Their model is trained on huge corpus with lots of GPU compute that is not feasible in an academia environment.

- Note that in Lee, Chang, and Toutanova (2019), dense retrieval outperformed BM25 in datasets such as NaturalQuestions and WebQuestions, but not TriviaQA or SQuAD. The authors hypothesize that in datasets like SQuAD or TriviaQA, the questions are elicited from annotators who already know the answer, hence creating a bias towards word overlap, whereas queries in NaturalQuestions and WebQuestions are derived from real users seeking information. Our dataset may suffer from similar biases.

Later work that discusses dense retrieval include Guu et al. (2020) and Karpukhin et al. (2020). Guu et al. (2020) uses the exact relevance function as ours (Equation 6.1), but since the model is updating, the index goes "stale" after every gradient update: their solution is to refresh the index by asynchronously re-embedding and re-indexing all documents every several hundred training iterations (Guu et al., 2020). This is not computationally feasible in an academic setting.

On the other hand, Karpukhin et al. (2020) achieves the feat of training the system on a single GPU, while also outperforms BM25 retrieval for open domain QA over Wikipedia. Again, their relevance is Equation 6.1 with BERT embeddings and FAISS for retrieval, and they do not need the massive ICT training used in Lee, Chang, and Toutanova (2019). Their key is a trick called "in-batch negative sampling", where in a batch $x_{0:n}$, the positive sample for x_i is a negative sample for x_j if $i \neq j$. The rationale is that for N questions, you get N^2 ranking pairs. Hence the batch size can be improved, thereby minimizing the variance of the gradients, achieving better performance. Additionally, they add only 1 negative sample from BM25 retrieval that is not gold. They found out that 1 is enough, no need for 2 (we used at least 10 negative samples from BM25 retrieval).

Additionally, recent work that I have collaborated in (Gao et al., 2020) points out that both lexical retrieval (based on sparse vector representations) and embedding-based retrieval (methods in this chapter that are based on dense representations) have merits and should be treated as complementary: one can train the neural vector embeddings so that they learn to capture semantics that lexical retrieval fails to capture, as a residual part of the lexical relevance. Empirical evaluation demonstrates the advantages of this method over various lexical retrieval models and a BERT-based dense embedding retrieval model, substantially narrowing the gap between full-collection retrieval and costly reranking systems, and setting a state-of-the-art result for *ad hoc* text retrieval. Later work should expand this combined method for **(1)** other retrieval tasks such as open-domain question answering, or claim verification; and **(2)** instead of using lexical features alone for the sparse representations, a more expressive feature system like what we proposed in Chapter 5 (T. Chen and Van Durme, 2017) can be utilized (e.g. to include named entity information) to boost the performance of these retrieval systems.

To summarize, we proposed dense vector based retrieval for QA and similar question retrieval in this chapter. While our experiment did not yield a competitive result against vanilla BM25 baselines or feature-augmented sparse vector based solutions in the previous chapter, later work in this thread shows that this idea can work with better pre-trained embeddings, larger distant-supervised pre-training, or better strategies in the creation of preference pairs for training.

Part III

Mentions and Entities

Chapter 7

Hierarchical Typing

In this chapter, we turn the focus from sentences to mentions in text, and study the task of *fine-grained entity typing*, i.e., assigning semantic types for mentions in text. We cast the problem as a type-ranking problem, where given a text mention, candidate types are ranked under a relevance function representing whether the mention is an instance of the specific type. These fine-grained entity typing systems found usages in various scenarios, such as task-oriented multi-domain dialogue understanding in agents such Siri or Alexa (T. Chen, Naik, et al., 2019), where various slots in dialogue sessions can be properly typed to aid downstream execution of these commands (e.g. playing music, or order a book).

The proposed method for hierarchical entity classification embraces the tree-structured ontology at both training and during prediction. At training, our novel multi-level learning-to-rank loss compares positive types against negative siblings according to the type tree. During prediction, we define a coarse-to-fine decoder that restricts viable candidates at each level of the ontology based on already predicted parent type(s). We achieve state-of-the-art across multiple datasets, particularly with respect to strict accuracy. Some materials in this chapter has been published in T. Chen, Y. Chen, and Van Durme (2020).

7.1 Introduction

Entity typing is the assignment of a semantic label to a span of text, where that span is usually a *mention* of some entity in the real world. Named entity recognition (NER) is a canonical information extraction task, commonly considered a form of entity typing that assigns spans to one of a handful of types, such as `PER` (person), `ORG` (organization), `GPE` (geo-political entity), and so on.

Fine-grained entity typing (FET) seeks to classify spans into types according to more diverse, semantically richer ontologies (Ling and Weld, 2012; Yosef et al., 2012; Dan Gillick et al., 2014; Del Corro et al., 2015; Choi et al., 2018), and has begun to be used in downstream models for entity linking (Gupta, Singh, and Roth, 2017; J. Raiman and O. Raiman, 2018).

Consider the example in Figure 7.1 from the FET dataset, FIGER (Ling and Weld, 2012). The mention of interest here in this example, *Hollywood Hills*, will be typed with the single label `LOC` in traditional NER, but may be typed with a *set* of types {`/location`, `/geography`, `/geography/mountain`} under a fine-grained typing scheme. In these finer-grained typing schemes, types usually form a hierarchy: there are a set of coarse

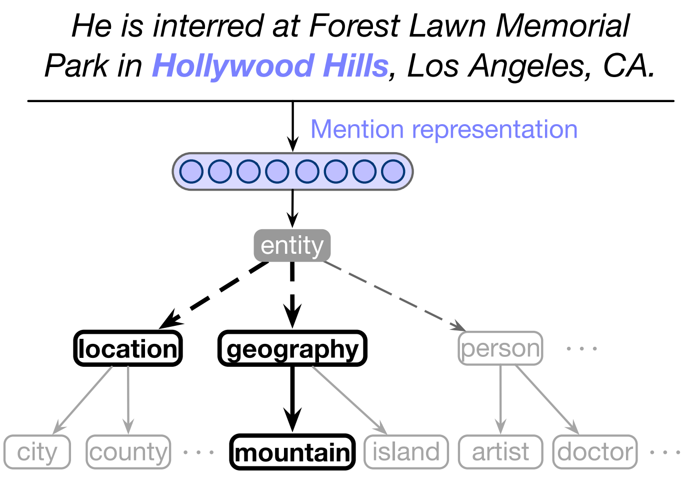


Figure 7.1: An example mention classified using the FIGER ontology. Positive types are bolded whereas negative types are grayed out.

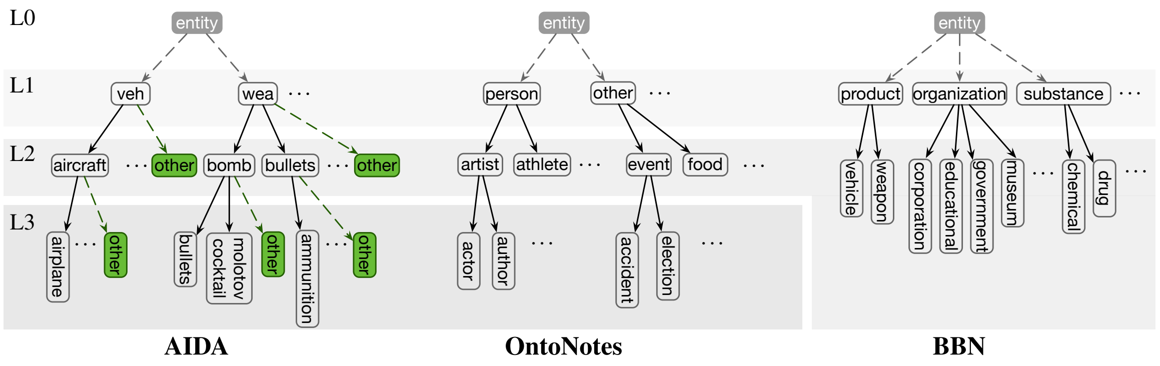


Figure 7.2: Various type ontologies. Different levels of the types are shown in different shades, from L0 to L3. The `ENTITY` and `OTHER` special nodes are discussed in Section 7.3.

types that lies on the top level—these are similar to traditional NER types, e.g. `/person`; additionally, there are finer types that are *subtypes* of these top-level types, e.g. `/person/artist` or `/person/doctor`.

Most prior work concerning fine-grained entity typing has approached the problem as a *multi-label classification* problem: given an entity mention together with its context, the classifier seeks to output a set of types, where each type is a node in the hierarchy.

Approaches to FET include hand-crafted sparse features to various neural architectures (Ren, W. He, Qu, L. Huang, et al., 2016; Shimaoka et al., 2017; Y. Lin and Ji, 2019) (for additional background introduction, see Section 7.2).

Perhaps owing to the historical transition from “flat” NER types, there has been relatively little work in FET that exploits the *tree structure* of the ontology, where type labels satisfy the *hierarchical property*: ***a subtype is valid only if its parent supertype is also valid.*** We propose a novel method that takes the explicit ontology structure into account, by a *multi-level learning to rank* approach that ranks the candidate types conditioned on the given entity mention. Intuitively, coarser types are easier whereas finer types are harder to classify: we capture this intuition by allowing distinct margins at each level of the ranking model. Coupled with a novel coarse-to-fine decoder that searches on the type hierarchy, our approach guarantees that predictions do not violate the hierarchical property, and achieves state-of-the-art results according to multiple measures across various commonly used datasets.

7.2 Related Work

FET is usually studied as allowing for sentence-level context in making predictions, notably starting with Ling and Weld (2012) and Dan Gillick et al. (2014), where they created the commonly used FIGER and OntoNotes datasets for FET. While researchers have considered the benefits of document-level (S. Zhang, Duh, and Van Durme, 2018) or corpus-level (Yaghoobzadeh and Schütze, 2015) context, here we focus on the sentence-level variant for best contrast to prior work.

Progress in FET has focused primarily on:

- **Better mention representations:** Starting from sparse hand-crafted binary features (Ling and Weld, 2012; Dan Gillick et al., 2014), the community has moved to distributed representations (Yogatama, Daniel Gillick, and Lazic, 2015), to pre-trained word embeddings with LSTMs (Ren, W. He, Qu, L. Huang, et al., 2016; Ren, W. He, Qu, Voss, et al., 2016; Shimaoka et al., 2016; Abhishek, Anand, and Awekar, 2017; Shimaoka et al., 2017) or CNNs (Murty et al., 2018), with mention-to-context attention (S. Zhang, Duh, and Van Durme, 2018), then to employing pre-trained language models like ELMo (Peters et al., 2018) to generate ever better representations (Y. Lin and Ji, 2019). Our approach builds upon these developments and uses state-of-the-art mention encoders.

- **Incorporating the hierarchy:** Most prior works approach the hierarchical typing problem as *multi-label classification*, without using information in the hierarchical structure, but there are a few exceptions. Ren, W. He, Qu, L. Huang, et al. (2016) proposed an adaptive margin for learning-to-rank so that similar types have a smaller margin; Xu and Barbosa (2018) proposed hierarchical loss normalization that penalizes output that violates the hierarchical property; and Murty et al. (2018) proposed to learn a *subtyping* relation to constrain the type embeddings in the type space. In contrast to these approaches, our coarse-to-fine decoding approach strictly guarantees that the output does not violate the hierarchical property, leading to better performance. HYENA (Yosef et al., 2012) applied ranking to sibling types in a type hierarchy, but the number of predicted positive types are trained separately with a meta-model, hence does not support neural end-to-end training.

Researchers have proposed alternative FET formulations whose types are not formed in a type hierarchy, in particular Ultra-fine entity typing (Choi et al., 2018; Xiong et al., 2019; Onoe and Durrett, 2019), with a very large set of types derived from phrases mined from a corpus. FET in KB (Jin et al., 2019) labels mentions to types in a knowledge base with multiple relations, forming a type graph. Dai et al. (2019) augments the task with entity linking to KBs.

7.3 Problem Formulation

We denote a mention as a tuple $x = (w, l, r)$, where $w = (w_1, \cdots, w_n)$ is the sentential context and the span $[l : r]$ marks a mention of interest in sentence w. That is, the mention of interest is $(w_l, \cdots, w_r)$. Given x, a hierarchical entity typing model outputs a set of types Y in the type ontology $\mathcal{Y}$, i.e. $Y \subseteq \mathcal{Y}$.

Type hierarchies take the form of a forest, where each tree is rooted by a top-level supertype (e.g. `/person`, `/location`, etc.). We add a dummy parent node `ENTITY` = "`/`", the supertype of all entity types, to all the top-level types, effectively transforming a type forest to a type tree. In Figure 7.2, we show 3 type ontologies associated with 3 different datasets (see Section 7.5.1), with the dummy `ENTITY` node augmented.

We now introduce some notation for referring to aspects of a type tree. The binary relation "type z is a subtype of y" is denoted as $z <: y$.[1] The unique parent of a type y in the type tree is denoted $\bar{y} \in \mathcal{Y}$, where $\bar{y}$ is undefined for $y = \texttt{ENTITY}$. The immediate subtypes of y (children nodes) are denoted $\text{Ch}(y) \subseteq \mathcal{Y}$. Siblings of y, those sharing the same immediate parent, are denoted $\text{Sb}(y) \subseteq \mathcal{Y}$, where $y \notin \text{Sb}(y)$.

In the AIDA FET ontology (see Figure 7.2), the maximum depth of the tree is $L = 3$, and each mention can only be typed with at most 1 type from each level. We term this scenario ***single-path*** typing, since there can be only 1 path starting from the root (`ENTITY`) of the type tree. This is in contrast ***multi-path*** typing, such as in the BBN dataset, where mentions may be labeled with multiple types on the same level of the tree.

We discuss the problem of *unspecified* subtypes. For example, in AIDA, there are mentions labeled such as as `/per/police/<unspecified>`. In FIGER, we find instances with labeled type `/person` but not any further subtype. What does it mean when a mention x is labeled with a ***partial type path***, i.e., a type y but none of the subtypes $z <: y$? We consider two interpretations:

- **Exclusive:** x is of type y, but x is not of any type $z <: y$.
- **Undefined:** x is of type y, but whether it is an instance of some $z <: y$ is unknown.

We devise different strategies to deal with these two conditions. Under the ***exclusive***

[1] Per programming language literature, e.g. the type system $F_{<:}$ that supports subtyping.

case, we add a dummy `OTHER` node to every intermediate branch node in the type tree. For any mention x labeled with type y but none of the subtypes $z <: y$, we add this additional label "y/`OTHER`" to the labels of x (see Figure 7.2: AIDA). For example, if we interpret a partial type path `/person` in FIGER as *exclusive*, we add another type `/person/OTHER` to that instance. Under the ***undefined*** case, we do not modify the labels in the dataset. We will see this can make a significant difference depending on the way a specific dataset is annotated.

7.4 Model

7.4.1 Mention Representation

Hidden representations for entity mentions in sentence w are generated by leveraging recent advances in language model pre-training, e.g. ELMo (Peters et al., 2018).[2] The ELMo representation for each token w_i is denoted as $\mathbf{w}_i \in \mathbb{R}^{d_w}$. Lexical dropout is applied with probability p_{D} to the ELMo vectors.

Our mention encoder largely follows Y. Lin and Ji (2019). First a mention representation is derived using the representations of the words in the mention. We apply a max

[2] Y. Lin and Ji (2019) found that ELMo performs better than BERT (Devlin et al., 2019) for FET. Our internal experiments also confirm this finding. We hypothesize that this is due to the richer character-level information contained in lower-level ELMo representations that are useful for FET.

pooling layer atop the mention after a linear transformation:[3]

$$\mathbf{m} = \mathrm{MaxPool}(\mathbf{T}\mathbf{w}_l, \cdots, \mathbf{T}\mathbf{w}_r) \in \mathbb{R}^{d_w} . \tag{7.1}$$

Then we employ mention-to-context attention first described in S. Zhang, Duh, and Van Durme (2018) and later employed by Y. Lin and Ji (2019): a context vector **c** is generated by attending the sentence with a query vector derived from the mention vector **m**. We use the multiplicative attention of Luong, Pham, and Manning (2015):

$$a_i \propto \exp(\mathbf{m}^{\mathrm{T}}\mathbf{Q}\mathbf{w}_i) \tag{7.2}$$

$$\mathbf{c} = \sum_{i=1}^{N} a_i \mathbf{w}_i \in \mathbb{R}^{d_w} \tag{7.3}$$

The final representation for an entity mention is generated via concatenation of the mention and context vector: $[\mathbf{m}\ ;\ \mathbf{c}] \in \mathbb{R}^{2d_w}$.

7.4.2 Type Scorer

We learn a type embedding $\mathbf{y} \in \mathbb{R}^{d_t}$ for each type $y \in \mathcal{Y}$. To score an instance with representation $[\mathbf{m}\ ;\ \mathbf{c}]$, we pass it through a 2-layer feed-forward network that maps into the same space as the type space $\mathbb{R}^{d_t}$, with tanh as the nonlinearity. The final score is an

[3] Y. Lin and Ji (2019) proposed an attentive pooler with a learned global query vector. We found out that a simple max pooling layer achieves similar performance.

inner product between the transformed feature vector and the type embedding:

$$F(x, y) = \mathrm{FFNN}([\mathbf{m}\ ;\ \mathbf{c}]) \cdot \mathbf{y}\,. \tag{7.4}$$

7.4.3 Hierarchical Learning-to-Rank

We introduce our novel hierarchical learning-to-rank loss that (1) allows for natural multi-label classification and (2) takes the hierarchical ontology into account.

We start with a multi-class hinge loss that ranks positive types above negative types (Weston and Watkins, 1999):

$$J_{\text{flat}}(x, Y) = \sum_{y \in Y} \sum_{y' \notin Y} [\xi - F(x, y) + F(x, y')]_+ \tag{7.5}$$

where $[x]_+ = \max\{0, x\}$. This is actually learning-to-rank with a ranking SVM (Joachims, 2002): the model learns to rank the positive types $y \in Y$ higher than those negative types $y' \notin Y$, by imposing a margin ξ between y and y': type y should rank higher than y' by ξ. Note that in Equation 7.5, since it is a linear SVM, the margin hyperparameter ξ could be just set as 1 (the type embeddings are linearly scalable), and we rely on L_2 regularization to constrain the type embeddings.

Multi-level Margins However, this method considers all candidate types to be *flat* instead of hierarchical — all types are given the same treatment without any prior on their relative position in the type hierarchy. Intuitively, coarser types (higher in the hierarchy) should be easier to determine (e.g. `/person` vs `/location` should be fairly easy for the model), but fine-grained types (e.g. `/person/artist/singer`) are harder.

We encode this intuition by **(i)** learning to rank types *only* on the same level in the type tree; **(ii)** setting different margin parameters for the ranking model with respect to different levels:

$$\sum_{y \in Y} \sum_{y' \in \mathrm{Sb}(y) \setminus Y} [\xi_{\mathrm{lev}(y)} - F(x, y) + F(x, y')]_+ \tag{7.6}$$

Here $\mathrm{lev}(y)$ denotes the level of the type y on the type tree: for example, $\mathrm{lev}(\texttt{ENTITY}) = 0$; $\mathrm{lev}(\texttt{/location}) = 1$, and $\mathrm{lev}(\texttt{/person/artist/singer}) = 3$. In Equation 7.6, each positive type y is only compared against its negative siblings $\mathrm{Sb}(y) \setminus Y$, and the margin hyperparameter is set to be $\xi_{\mathrm{lev}(y)}$, i.e., a margin dependent on which level y is in the tree. Intuitively, we should set $\xi_1 > \xi_2 > \xi_3$ since our model should be able to learn a larger margin between easier pairs: we show that this is superior than using a single margin in our experiments.

Analogous to the reasoning that in Equation 7.5 the margin ξ can just be 1, only the relative ratios between ξ's are important. For simplicity,[4] if the ontology has L levels, we assign

$$\xi_l = L - l + 1 \, . \tag{7.7}$$

[4] We did hyperparameter search on these margin hyperparameters and found that Equation 7.7 generalized well.

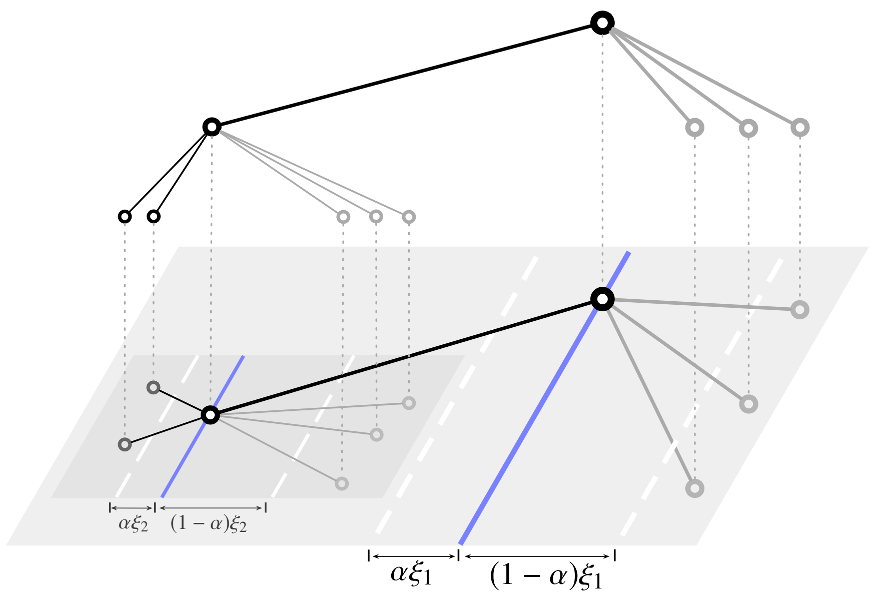


Figure 7.3: Hierarchical learning-to-rank. Positive type paths are colored black, negative type paths are colored gray. Each blue line corresponds to a threshold derived from a parent node. Positive types (on the left) are ranked above negative types (on the right).

For example, given an ontology with 3 levels, the margins per level are $(\xi_1, \xi_2, \xi_3) = (3, 2, 1)$.

Flexible Threshold Equation 7.6 only ranks positive types higher than negative types so that all children types given a parent type are ranked based on their relevance to the entity mention. What should be the threshold between positive and negative types? We could set the threshold to be 0 (approaching the multi-label classification problem as a set of binary classification problem, see Y. Lin and Ji (2019)), or tune an adaptive, type-specific threshold for each parent type (S. Zhang, Duh, and Van Durme, 2018). Here, we propose a simpler method.

We propose to directly use *the parent node as the threshold*. If a positive type is y, we learn the following ranking relation:

$$y \succ \bar{y} \succ y', \quad \forall y' \in \mathrm{Sb}(t) \tag{7.8}$$

where $\succ$ means “precedes”, or “ranks higher than”. For example, a mention has gold type `/person/artist/singer`. Since the parent type `/person/artist` can be considered as a kind of *prior* for all types of artists, the model should learn that the positive type “singer” should have a higher confidence than “artist”, and in turn, higher than other types of artists like “author” or “actor”. Hence the ranker should learn that “a positive subtype should rank higher than its parent, and its parent should rank higher than its negative children.” Under this formulation, at decoding time, given parent type y, a child subtype $z <: y$ that scores higher than y should be output as a positive label.

We translate the ranking relation in Equation 7.8 into a ranking loss that extends Equation 7.6. In Equation 7.6, there is an expected margin ξ between positive types and negative types. Since we inserted the parent in the middle, we divide the margin ξ into $\alpha\xi$ and $(1-\alpha)\xi$: $\alpha\xi$ being the margin between positive types and the parent; and $(1-\alpha)\xi$ is the margin between the parent and the negative types. For a visualization see Figure 7.3.

The hyperparameter $\alpha \in [0, 1]$ can be used to tune the precision-recall tradeoff when outputting types: the smaller α, the smaller the expected margin there is between positive types and the parent. This intuitively increases precision but decreases recall (only very confident types can be output). Vice versa, increasing α decreases precision but increase recall. We present a diagnosis of how α influences precision and recall in the experiment section below.

Therefore we learn 3 sets of ranking relations from Equation 7.8:

- Positive types should be scored above parent by $\alpha\xi$:

$$J_{y>\bar{y}} = [\alpha\xi_{\text{lev}(y)} - F(x, y) + F(x, \bar{y})]_+ \tag{7.9}$$

- Parent should be scored above any negative sibling types by $(1 - \alpha)\xi$:

$$J_{\bar{y}>y'} = \sum_{y' \in \text{Sb}(y) \setminus Y} [(1 - \alpha)\xi_{\text{lev}(y)} - F(x, \bar{y}) + F(x, y')]_+ \tag{7.10}$$

- Positive types should be scored above negative sibling types by ξ:

$$J_{y>y'} = \sum_{y' \in \text{Sb}(y) \setminus Y} [\xi_{\text{lev}(y)} - F(x, y) + F(x, y')]_+ \tag{7.11}$$

Our final hierarchical ranking loss is formulated as the sum of the 3 components listed above.

$$J_{\text{hier}}(x, Y) = \sum_{y \in Y} \left(J_{y>\bar{y}} + J_{\bar{y}>y'} + J_{y>y'}\right) \tag{7.12}$$

7.4.4 Decoding

Predicting the types for each entity mention can be performed via iterative searching on the type tree, from the root `ENTITY` node to coarser types, then to finer-grained types. This ensures that our output does not violate the hierarchical property, i.e., if a subtype is output, its parent must be output.

Given instance x we compute the score $F(x, y)$ for each type $y \in \mathcal{Y}$, the searching process starts with the root node `ENTITY` of the type tree in the queue. For each type y in the node, a child node $z <: y$ (subtypes) is added to the predicted type set if $F(x, z) > F(x, y)$, corresponding to the ranking relation in Equation 7.8 that the model has learned.[5]

Here we only take the top-k element to add to the queue to prevent from an over-generation of types. This can also be used to enforce the single-path property (setting $k = 1$) if the dataset is single-path. For each level i in the type hierarchy, we limit the branching factor (allowed children) to be k_i. The algorithm is listed in Algorithm 1, where the function $\textsc{TopK}(S, k, f)$ selects the top-k elements from S with respect to the function f.

[5] For the OntoNotes dataset, we introduce another set of per-level hyperparameters $\delta_{\text{lev}(y)}$, and the threshold value $F(x, y)$ is modified to $F(x, y) + \delta_{\text{lev}(y)}$, akin to the adaptive threshold in S. Zhang, Duh, and Van Durme (2018). This is due to a large type distribution mismatch between the training and dev/test sets in OntoNotes (in dev/test there are a lot of instances with the single type `/other` but not in the training set). For other datasets they are unused, i.e. just 0.

Algorithm 1 Decoding for Hierarchical Typing

```
 1: function HierTypeDec($F(x, \cdot)$)
 2:     $Q \leftarrow \{\texttt{ENTITY}\}$                                   ▷ queue for searching
 3:     $\hat{Y} \leftarrow \varnothing$                                     ▷ set of output types
 4:     repeat
 5:         $y \leftarrow \textsc{Dequeue}(Q)$
 6:         $\theta \leftarrow F(x, y) + \delta_{\text{lev}(y)}$                        ▷ threshold value
 7:         $Z \leftarrow \{z \in \text{Ch}(y) \mid F(x, z) > \theta\}$          ▷ all decoded children types
 8:         $Z' \leftarrow \textsc{TopK}(Z, k_{\text{lev}(y)+1}, F(x, \cdot))$   ▷ pruned by the max branching factors
 9:         $\hat{Y} \leftarrow \hat{Y} \cup Z'$
10:         for $z \in Z'$ do
11:             $\textsc{Enqueue}(Q, z)$
12:         end for
13:     until $Q = \varnothing$                                              ▷ queue is empty
14: return $\hat{Y}$                                                          ▷ return all decoded types
15: end function
```

7.4.5 Subtyping Relation Constraint

Each type $y \in \mathcal{Y}$ in the ontology is assigned a type embedding $\mathbf{y} \in \mathbb{R}^{d_t}$. We notice the binary subtyping relation " $<:$ " $\subseteq \mathcal{Y} \times \mathcal{Y}$ on the types. Trouillon et al. (2016) proposed the relation embedding method ComplEx that works well with anti-symmetric and transitive relations such as subtyping. It has been employed in FET before — in Murty et al. (2018), ComplEx is added to the loss to regulate the type embeddings. ComplEx operates in the complex space — we use the natural isomorphism between real and complex spaces to map the type embedding into complex space (first half of the embedding vector as the real part, and the second half as the imaginary part):

$$\phi : \mathbb{R}^{d_t} \to \mathbb{C}^{d_t/2} \tag{7.13}$$

$$\mathbf{t} = [\ \text{Re}\,\phi(\mathbf{t})\ ;\ \text{Im}\,\phi(\mathbf{t})\ ] \tag{7.14}$$

We learn a single relation embedding $\mathbf{r} \in \mathbb{C}^{d_t/2}$ for the subtyping relation. Given type y and z, the subtyping statement $y <: z$ is modeled using the following scoring function:

$$r(y, z) = \mathrm{Re}\Big(\mathbf{r} \cdot \Big(\phi(\mathbf{y}) \odot \overline{\phi(\mathbf{z})}\Big)\Big) \tag{7.15}$$

where $\odot$ is element-wise product and $\bar{x}$ is the complex conjugate of x. If $y <: z$ then $r(y, z) > 0$; and vice versa, $r(y, z) < 0$ if $y \not<: z$.

Loss Given instance (x, Y), for each positive type $y \in Y$, we learn the following relations:

$$\begin{aligned} & y <: \bar{y} \\ & y \not<: y', \quad \forall y' \in \mathrm{Sb}(y) \\ & y \not<: y', \quad \forall y' \in \mathrm{Sb}(\bar{y}) \end{aligned} \tag{7.16}$$

Translating these relation constraints as a binary classification problem ("is or is not a subtype") under a primal SVM, we get a hinge loss:

$$J_{\mathrm{rel}}(x, Y) = \sum_{y \in Y} \left([1 - r(y, \bar{y})]_+ + \sum_{y' \in \mathrm{Sb}(y) \cup \mathrm{Sb}(\bar{y})} [1 + r(y, y')]_+ \right). \tag{7.17}$$

This is different from Murty et al. (2018), where a binary cross-entropy loss on randomly sampled (y, y') pairs is used. Our experiments showed that the loss in Equation 7.17

performs better than the cross-entropy version, due to the structure of the training pairs: we use siblings and siblings of parents as negative samples (these are types closer to the positive parent type), hence are training with more competitive negative samples.

7.4.6 Training and Validation

Our final loss is a combination of the hierarchical ranking loss and the subtyping relation constraint loss, with L_2 regularization over all parameters $\mathbf{\Theta}$:

$$J_{\text{hier}}(x, Y) + \beta J_{\text{rel}}(x, Y) + \frac{\lambda}{2} \|\mathbf{\Theta}\|_2^2 \,. \tag{7.18}$$

The AdamW optimizer (Loshchilov and Hutter, 2019) is used to train the model, as it is shown to be superior than the original Adam under L_2 regularization. Hyperparameters α (ratio of margin above/below threshold), β (weight of subtyping relation constraint), and λ (L_2 regularization coefficient) are tuned.

At validation time, we tune the maximum branching factors for each level $k_1, \cdots, k_L$.[6] These parameters tune the trade-off between the precision and recall for each layer and prevents over-generation (as we observed in some cases). All hyperparameters are tuned so that models achieve maximum micro F_1 scores (see Section 7.5.4).

[6] For the OntoNotes dataset, this also includes the per-level threshold $\delta_{\text{lev}(k)}$.

Dataset	**Train**	**Dev**	**Test**	**# Levels**	**# Types**	**Multi-path?**
AIDA	2,492	558	1,383	3	187	single-path
BBN	84,078	2,000	13,766	2	56	multi-path
OntoNotes	251,039	2,202	8,963	3	89	multi-path
FIGER	2,000,000	10,000	563	2	113	multi-path

Table 7.1: Statistics of various datasets.

7.5 Experiments

7.5.1 Datasets

AIDA The AIDA Phase 1 practice dataset for hierarchical entity typing comprises of 297 documents from `LDC2019E04` / `LDC2019E07`, and the evaluation dataset is from `LDC2019E42` / `LDC2019E77`. We take only the English part of the data, and use the practice dataset as train/dev, and the evaluation dataset as test. The practice dataset comprises of 3 domains, labeled as `R103`, `R105`, and `R107`. Since the evaluation dataset is out-of-domain, we use the smallest domain `R105` as dev, and the remaining `R103` and `R107` as train.

The AIDA entity dataset has a 3-level ontology, termed *type*, *subtype*, and *subsubtype*. A mention can only have one label for each level, hence the dataset is *single-path*, thus the branching factors (k_1, k_2, k_3) for the three layers are set to $(1, 1, 1)$.

BBN Weischedel and Brunstein (2005) labeled a portion of the one million word Penn Treebank corpus of Wall Street Journal texts (`LDC95T7`) using a two-level hierarchy, re-

Dataset	α	β	λ	p_{D}	$k_{1,\cdots,L}$
AIDA	0.1	0.3	0.1	0.6	(1,1,1)
BBN	0.2	0.1	0.003	0.5	(2,1)
OntoNotes	0.15	0.1	0.001	0.5	(2,1,1)
FIGER	0.2	0.1	0.0001	0.5	(2,1)

Table 7.2: Hyperparameters tuned for these datasets.

sulting in the BBN Pronoun Coreference and Entity Type Corpus. We follow the train/test split by Ren, W. He, Qu, Voss, et al. (2016), and follow the train/dev split by S. Zhang, Duh, and Van Durme (2018).

OntoNotes Dan Gillick et al. (2014) sampled sentences from the OntoNotes corpus and annotated the entities using 89 types. We follow the train/dev/test data split by Shimaoka et al. (2017).

FIGER Ling and Weld (2012) sampled a dataset from Wikipdia articles and news reports. Entity mentions in these texts are mapped to a 113-type ontology derived from Freebase (Bollacker et al., 2008). Again, we follow the data split by Shimaoka et al. (2017).

The statistics of these datasets and their accompanying ontologies are listed in Table 7.1.

7.5.2 Setup

To best compare to recent prior work, we follow Y. Lin and Ji (2019) where the ELMo encodings of words are fixed and not updated. We use all 3 layers of ELMo output, so the initial embedding has dimension $d_w = 3072$. We set the type embedding dimensionality to be $d_t = 1024$. The initial learning rate is 10^{-5} and the batch size is 256.

Hyperparameter choices are tuned on dev sets, and are listed in Table 7.2.[7] We employ early stopping: choosing the model that yields the best micro F_1 score on dev sets.

Our models are implemented using AllenNLP (Gardner et al., 2018), with implementation for subtyping relation constraints from OpenKE (Han et al., 2018).

7.5.3 Baselines

We compare our approach to major prior work in FET that are capable of *multi-path* entity typing.[8] For AIDA, since there are no prior work on this dataset to our knowledge, we also implemented multi-label classification as set of binary classifier models (similar to Y. Lin and Ji (2019)) as a baseline, with our mention feature extractor. The results are shown in Table 7.3 as "Multi-label".

7.5.4 Metrics

We follow prior work and use strict accuracy (Acc), macro F_1 (MaF), and micro F_1 (MiF) scores. Given instance x_i, we denote the gold type set as Y_i and the predicted type set $\hat{Y}_i$. The strict accuracy is the ratio of instances where $Y_i = \hat{Y}_i$. Macro F_1 is the average of all

[7] The OntoNotes dataset has an additional set of hyperparameters, i.e. the per-level threshold $\delta_{1,2,3} = (2.5, 3.0, 0.0)$.

[8] S. Zhang, Duh, and Van Durme (2018) included document-level information in their best results—for fair comparison, we used their results without document context, as are reported in their ablation tests.

F_1 scores between Y_i and $\hat{Y}_i$ for all instances, whereas micro F_1 counts total true positives, false negatives and false positives globally.

We also investigate per-level accuracies on AIDA. The accuracy on level l is the ratio of instances whose predicted type set and gold type set are identical at level l. If there is no type output at level l, we append with `OTHER` to create a dummy type at level l: e.g. `/person/OTHER/OTHER`. Hence accuracy of the last level (in AIDA, level 3) is equal to the strict accuracy.

7.5.5 Results

All our results are run under the two conditions regarding partial type paths: exclusive or undefined. The result of the AIDA dataset is shown in Table 7.3. Our model under the exclusive case outperforms a multi-label classification baseline over all metrics.

Of the 187 types specified in the AIDA ontology, the train/dev set only covers 93 types. The test set covers 85 types, of which 63 are seen types. We could perform zero-shot entity typing by initializing a type's embedding using the type name (e.g. `/fac/structure/plaza`) together with its description (e.g. "*An open urban public space, such as a city square*") as is designated in the data annotation manual. We leave this as future work.

Results for the BBN, OntoNotes, and FIGER can be found in Table 7.4. Across 3 datasets, our method produces the state-of-the-art performance on strict accuracy and micro F_1 scores, and state-of-the-art or comparable ($\pm 0.5\%$) performance on macro F_1 score, as

Approach	L1	L2	L3	MaF	MiF
Ours (exclusive)	**81.6**	43.1	**32.0**	**60.6**	**60.0**
Ours (undefined)	80.0	**43.3**	30.2	59.3	58.0
– Subtyping constraints	80.3	40.9	29.9	59.1	58.3
– Multi-level margins	76.9	40.2	29.8	57.4	56.9
Multi-label	80.5	42.1	30.7	59.7	57.9

Table 7.3: Results on the AIDA dataset.

Approach	**BBN**			**OntoNotes**			**FIGER**		
	Acc	MaF	MiF	Acc	MaF	MiF	Acc	MaF	MiF
Ling and Weld (2012)	46.7	67.2	61.2		– [†]		52.3	69.9	69.3
Ren, W. He, Qu, Voss, et al. (2016)	49.4	68.8	64.5	51.6	67.4	62.4	49.4	68.8	64.5
Ren, W. He, Qu, L. Huang, et al. (2016)	67.0	72.7	73.5	55.1	71.1	64.7	53.3	69.3	66.4
Abhishek, Anand, and Awekar (2017)	60.4	74.1	75.7	52.2	68.5	63.3	59.0	78.0	74.9
Shimaoka et al. (2017)		– [†]		51.7	71.0	64.9	59.7	79.0	75.4
Murty et al. (2018)		– [†]			– [†]		59.7	78.3	75.4
S. Zhang, Duh, and Van Durme (2018)	58.1	75.7	75.1	53.2	72.1	66.5	*60.2*[‡]	*78.7*[‡]	*75.5*[‡]
Y. Lin and Ji (2019)	55.9	79.3	78.1	*63.8*[*]	*82.9*[*]	*77.3*[*]	62.9	**83.0**	79.8
Ours (exclusive)	48.2	63.2	61.0	58.3	72.4	67.2	**69.1**	82.6	**80.8**
Ours (undefined)	**75.2**	**79.7**	**80.5**	**58.7**	**73.0**	**68.1**	65.5	80.5	78.1
– Subtyping constraint	73.2	77.8	78.4	58.3	72.2	67.1	65.4	81.4	79.2
– Multi-level margins	68.9	73.2	74.2	58.5	71.7	66.0	68.1	80.4	78.0

[†]: Not run on the specific dataset; [*]: Not strictly comparable due to non-standard, much larger training set;
[‡]: Result has document-level context information, hence not comparable.

Table 7.4: Results of common FET datasets: BBN, OntoNotes, and FIGER. Numbers in italic are results obtained with various augmentation techniques, either larger data or larger context, hence not directly comparable.

compared to prior models, e.g. (Y. Lin and Ji, 2019). Especially, our method improves upon the strict accuracy substantially (4%–8%) across these datasets, showing our decoder are better at outputting exact correct type sets.

7.5.6 Discussions

Effect of the threshold ratio hyperparameter In the model described above, we introduced a hyperparameter α that can be used to tune the precision-recall tradeoff when outputting types: the smaller α is, we expect the precision would be higher, and vice versa, the greater α is, the recall would be higher. We show evidence that supports this hypothesis under the BBN dataset, with α taking the values in $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$ (Figure 7.4). Clearly, we see that with increasing α the precision drops but the recall improves. We hence tune α as a hyperparameter to achieve the highest overall F_1 score.

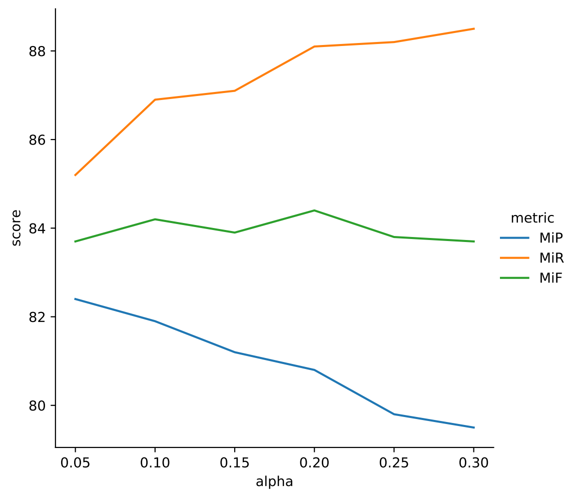


Figure 7.4: The micro precision (MiP), recall (MiR), and F_1 score (MiF) with varying threshold ratio α on the BBN dev set.

Partial type paths: exclusive or undefined? Interestingly, we found that for AIDA and FIGER, partial type paths should be better considered as *exclusive*, whereas for BBN and OntoNotes, considering them as *undefined* leads to better performance. We hypothesize that this comes from how the data is annotatated—the annotation manual may contain directives as whether to interpret partial type paths as exclusive or undefined, or the data may be non-exhaustively annotated, leading to undefined partial types. We advocate for careful investigation into partial type paths for future experiments and data curation.

Ablation Studies We compare our best model with various components of our model removed, to study the gain from each component. From the best of these two settings (*exclusive* and *undefined*), we report the performance of (i) removing the subtyping constraint as is described in Section 7.4.5; (ii) substituting the multi-level margins in Equation 7.7 with a "flat" margin, i.e., margins on all levels are set to be 1. These results are shown in Table 7.3 and Table 7.4 under our best results, and they show that both multi-level margins and subtyping relation constraints offer orthogonal improvements to our models.

Error Analyses We identify common patterns of errors, coupled with typical examples:

- Confusing similar types: In BBN, our model outputs `/gpe/city` when the gold type is `/location/region` for "... *in shipments from the* $\underline{\textit{Valley}}$ *of either hardware or software goods.*" These types are semantically similar, and our model failed to discriminate between these types.

- Missing correct types: There could be mentions like "*UW Department of Chemistry*" labeled as `/organization` but missing another correct type `/education/department`.

- Prediction too coarse: In FIGER, given instance "... *multi-agency investigation headed by the U.S. Immigration and Customs Enforcement 's homeland security investigations unit*", the gold types are `/government_agency` and `/organization`, but our model failed to output `/organization`, resulting in an incomplete type path.

- Prediction too fine: Given instance "*...said professor Charles Campbell, who was elected as an Fellow in 2010*", the mention "*Charles Campbell*" is labeled as `/person` in the gold label but our model outputs `/person/author`. One could argue that a professor is an author who writes papers, but this is not annotated as such in the FIGER dataset. This is an instance of our model making too fine predictions rather than too coarse.

- Focused on only parts of the mention: In AIDA, given instance "... *suggested they were the work of Russian special forces assassins out to blacken the image of Kievs pro-Western authorities*", our model outputs `/org/government` whereas the gold type is `/per/militarypersonnel`. Our model focused on the "Russian special forces" part, but ignored the "assassins" part. Better mention representation is required to correct this, possibly by introducing type-aware mention representation—we leave this as future work.

We manually sample and inspect 100 predictions errors in the FIGER dev set, and classifies these errors into bins to show what kinds errors our models make:

Error type	Proportion
Confusing similar types	15%
Missing correct types	22%
Prediction too coarse	15%
Prediction too fine	35%
Focused on only parts of the mention	6%
Totally wrong	7%

Table 7.5: Common types of errors of our hierarchical entity typing system.

7.6 Experiments for UltraFine

7.6.1 Task and Dataset

Ultra-fine entity typing (Choi et al., 2018) is a newly proposed task for entity typing: instead of a relatively coarse tree-structured ontology with about 100 types (see Table 7.1), in ultra-fine entity typing, the goal is to predict a set of free-form noun phrases as very fine-grained types (Choi et al., 2018). In their version, the ontology has more than 10,000 types (we will call the ontology and the accompanying dataset simply as "UltraFine" from now on).

Their UltraFine types can be classified into 3 disjoint bins:

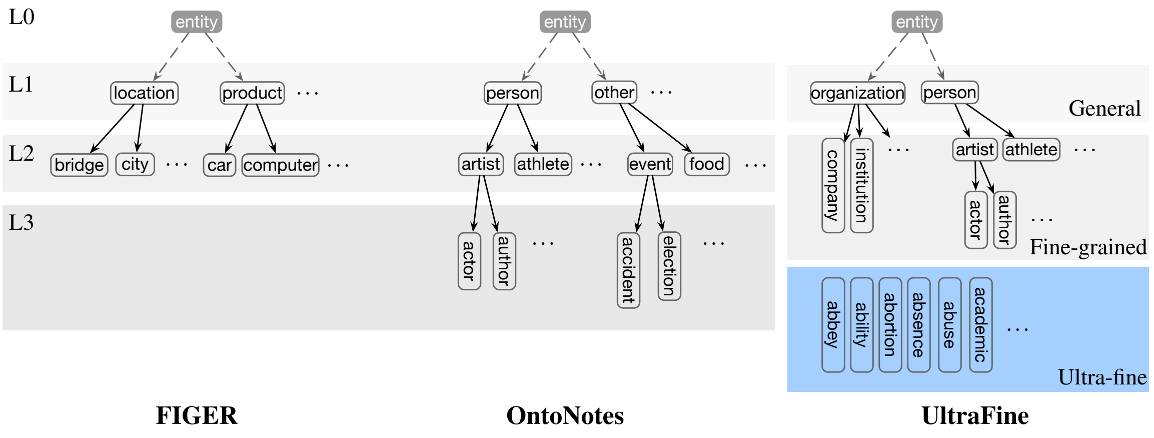


Figure 7.5: The typing ontology of UltraFine. Its top two levels (general and fine-grained) are formed by taking the union of the ontologies of FIGER and OntoNotes, but included a new *ultfa-fine* third level (colored blue).

- **General**: These 9 types include `person`, `location`, `object`, `organization`, `place`, `entity`, `object`, `time`, `event`;
- **Fine-grained**: These 121 types are formed by taking the union of existing type ontologies FIGER (Ling and Weld, 2012) and OntoNotes (Dan Gillick et al., 2014);
- **Ultra-fine**: These 10,201 types are mined by Choi et al. (2018), and encompasses a very large space.

The task for UltraFine is similar to our aforementioned fine-grained entity typing task: it evaluated as a multi-label classification task. The difference here is that instead of a predefined tree-structured ontology, the UltraFine ontology only assumes 3 levels without a well-defined tree structure.

The original dataset provided by Choi et al. (2018) contains a high quality crowd-sourced subset, and a distantly supervised augmented dataset. This dataset is further fil-

Dataset	Train (with augmented data)	Dev	Test
Original UltraFine (Choi et al., 2018)	1,998 (aug. 6,224,985)	1,998	1,998
Denoised (Onoe and Durrett, 2019)	1,998 (aug. 726,890)	1,998	1,998

Table 7.6: Statistics for various UltraFine entity typing datasets.

tered and denoised by Onoe and Durrett (2019), which leads to better typing performance than the original dataset, and is what we are going to use throughout this section. Statistics of these datasets can be found in Table 7.6.

7.6.2 Hierarchical Models

Since the *fine-grained* subset of the type ontology is derived from tree-structured ontologies FIGER and OntoNotes, we can restore their tree structure from FIGER and OntoNotes. We manually connect these nodes to their semantic hypernyms in the *general* type subset in UltraFine, resulting in a tree structure for the *general* and *fine-grained* subsets with depth 3 (see Figure 7.5).

Hence the task of ultra-fine typing is decomposed to two subtasks: **(1)** A *hierarchical* part dealing with the top 2 levels of the UltraFine ontology; and **(2)** a *flat* part dealing with the 3rd (i.e. ultra-fine) level of the UltraFine ontology.

For the *hierarchical* part, the model described in previous sections is used. For the *flat* type, we also employ a ranking-based solution similar to the solution we discussed, where we add an additionally dummy type ε for these flat types. The model learns the ranking relation that any positive ultra-fine type ranks above the dummy type ε, and the dummy

Algorithm 2 Decoding for Flat Typing

```
1: function FLATTYPEDEC($F(x, \cdot)$)
2:     $\theta \leftarrow F(x, \varepsilon)$                                      ▷ threshold value
3:     $Z \leftarrow \{z \in \mathcal{Y}_{\text{flat}} \mid F(x, z) > \theta\}$            ▷ all decoded children types
4:     $Z' \leftarrow \text{TopK}(Z, k_{\text{flat}}, F(x, \cdot))$      ▷ pruned by the max branching factor
5: return $Z'$                                           ▷ return all decoded types
6: end function
```

type ε in turn ranks above any negative ultra-fine type. This can be formulated as

$$J_{y>\varepsilon} = [\alpha\xi_{\text{flat}} - F(x, y) + F(x, \varepsilon)]_+ \tag{7.19}$$

$$J_{\varepsilon>y'} = \sum_{y' \notin Y} [(1 - \alpha)\xi_{\text{flat}} - F(x, \varepsilon) + F(x, y')]_+ \tag{7.20}$$

$$J_{y>y'} = \sum_{y' \notin Y} [\xi_{\text{flat}} - F(x, y) + F(x, y')]_+ \tag{7.21}$$

$$J_{\text{flat}}(x, Y) = J_{y>\varepsilon} + J_{\varepsilon>y'} + J_{y>y'} \tag{7.22}$$

The model is learned as a multitask problem, where the loss is a weighted sum of the hierarchical loss for the general and fine-grained levels of UltraFine, and the flat loss (Equation 7.22) for the ultra-fine level.

To decode, we adopt Algorithm 1 for a single-level setting.

Note that the newly included hyperparameters here: ξ_{flat} is the margin hyperparameter for flat types; and k_{flat} is the maximum number of positive types that can be decoded on the ultra-fine level.

7.6.3 Setup

We use the filtered UltraFine dataset from Onoe and Durrett (2019). Baselines include the original Choi et al. (2018) dataset, which used a GloVe-based LSTM encoder; and Onoe and Durrett (2019) using both the original dataset and the filtered dataset. Both of the results from Onoe and Durrett (2019) uses ELMo Peters et al. (2018) as the encoder.

The Onoe and Durrett (2019) model is similar with ours with respect to the encoder, where they also utilized ELMo-based embeddings. However, they model the problem as a set of binary classifiers for each type with a neural function $F(x, y) = \sigma(\mathrm{FFNN}_{\boldsymbol{\theta}}(\mathbf{x}) \cdot \mathbf{y})$. This is essentially a pointwise relevance function without any pairwise ranking.

From the intuition we discussed that coarser types should have larger margins and finer types should have smaller margins, we set the margin hyperparameters for the hierarchical and the flat levels aspects

$$(\xi_1, \xi_2, \xi_3, \xi_{\text{flat}}) = (4, 3, 2, 1) \ , \tag{7.23}$$

where $\xi_{1,2,3}$ corresponds to the 3 levels of the union of FIGER and OntoNotes, that covers the general and fine-grained levels in UltraFine.

For other hyperparameters, we select the maximum branching factors for the hierarchy $(k_1, k_2, k_3) = (2, 2, 1)$, for flat types $k_{\text{flat}} = 8$ (this means that at most 8 ultra-fine types

Dataset	General	Fine	Ultra-fine	Overall
Original Choi et al. (2018)	61.0	39.4	14.6	31.3
Onoe and Durrett (2019) w/ original data	70.7	42.2	17.1	35.7
Onoe and Durrett (2019) w/ filtered data	**73.2**	**43.8**	25.2	40.1
Ours	72.5	42.8	**29.2**	**41.5**

Table 7.7: Results for UltraFine entity typing. All numbers are micro-F_1 scores.

will be predicted[9]), threshold ratio $\alpha = 0.1$, relation constraint coefficient $\beta = 0.1$, L_2 regularization coefficient $\lambda = 0.0003$ and dropout rate $p_{\mathrm{D}} = 0.3$.

We follow Onoe and Durrett (2019) for the metrics: the micro-F_1 scores for the three levels (general, fine-grained, and ultra-fine) are reported, together with the overall micro-F_1 score.

7.6.4 Results and Discussions

The results for the UltraFine entity typing task can be found in Table 7.7.

Our ranking-based model achieves a state-of-the-art results over Onoe and Durrett (2019). Especially, the 4% gain on the flat, ultra-fine type level demonstrates the effectiveness of the ranking approach than the set of classifiers approach. We hypothesize that the slight performance drop on the hierarchical part is due to insufficient hyperparameter searching: the hyperparameters used are empirically chosen based on experiments on OntoNotes and FIGER, and they might not reflect the best achievable performance on the

[9] This is based on the observation that the maximum number of annotated types in the dev set is 8.

Choi et al. (2018) UltraFine datset since we had not extensively searched the hyperparameter space.

7.7 Conclusions

We proposed **(i)** a novel multi-level learning to rank loss function that operates on a type tree, and **(ii)** an accompanying coarse-to-fine decoder to fully embrace the ontological structure of the types for hierarchical entity typing. Our approach achieved state-of-the-art performance across various datasets, and made substantial improvement (4–8%) upon strict accuracy.

Additionally, we advocate for careful investigation into *partial type paths*: their interpretation relies on how the data is annotated, and in turn, influences typing performance.

We also extended the ranking-based method to the UltraFine entity typing task, where the type ontology has a hierarchical subset and also a “flat”, unstructured subset. Proof-of-concept experiments demonstrate that ranking-based method indeed leads to improvements under the UltraFine setting.

Chapter 8

Cross-document Coreference Resolution

In this chapter we turn the focus to the semantic relevance of mention *coreference* (i.e. referring to the same entity), and explore how we can reuse a ranking function trained for recognizing in-document coreference resolution for retrieval across multiple documents. We devise methods to extrapolate relevance functions learned under the in-document scenario to a *cross-document* scenario to *retrieve* possible coreferent mentions, enabling coreference resolvers to attend not only just to in-document antecents, but also to mentions in other documents.

We explore **(1)** substituting a marginal log likelihood loss to a learning-to-rank loss for candidate antecedents, and **(2)** extending the second-order inference algorithm in the in-document coreference resolution to a cross-document scenario, executing under the theme of triage-then-rerank of this thesis. Methods proposed in this chapter can be reused for other scenarios that calls for attention mechanisms spanning over massive collections of elements.

8.1 Introduction

Coreference resolution performed on noun phrases (NPs), the task of determining which NPs in a document or dialogue refer to the same real-world entity, has long been at the core of NLP (Ng, 2010). The task is commonly considered under two different scenarios: **(1)** *in-document* coreference resolution, where the scope of the text is one document or one dialogue session; or **(2)** *cross-document* coreference resolution, where a system should predict links between NPs not only in the same document, but also across different documents in a corpus.

In-document coreference resolution is related to the task of anaphora resolution, whose goal is to identify an *antecedent* for an *anaphoric* NP (e.g. "*her*", i.e., an NP that depends on an antecedent NP for its semantic interpretation) (van Deemter and Kibble, 2000). Coreference resolution is considered as a hard NLP task, since correctly identifying coreference chains involves sophisticated commonsense knowledge and inference procedures (Charniak, 1972). Indeed, for challenge coreference resolution datasets such as the Winograd Schema Challenge (WSC) (Levesque, 2011; Levesque, Davis, and Morgenstern, 2012) or more recently WINOGRANDE (Sakaguchi, Le Bras, et al., 2020), tasks proposed as alternative Turing tests, state-of-the-art NLP models still lag behind human performance by a large margin.

There has been abundant work on in-document coreference resolution, either based on traditional syntactic features, or recently pure neural learning methods. State-of-the-art methods include a *span-ranking* module that decides, for each span, which of the previous spans (if any) is a good antecedent. It involves a span-pair model that given a span x, produces a relevance score for any antecedent span x' that represents how likely x' is an antecedent of span x. Additionally, *higher-order* coreference resolution has proven to be superior (Lee, L. He, and Zettlemoyer, 2018), where span representations are refined by *attending* to other previous mentions, enabling the model to softly condition on predicted clusters instead of prior mentions.

This falls under the overarching theme of this thesis of ranking candidates (here, antecedent spans) given a query (here, an NP span) under a semantic relevance function (here, the semantic relation of coreference). We devise methods to extrapolate relevance functions learned under the in-document scenario to a *cross-document* scenario to *retrieve* possible coreferent mentions. This retrieval is a natural generalization of the higher-order attention described above by being an approximate corpus-wide attention instead of in-document attention. We investigate whether this corpus-wide retrieval can bring improvements to in-document coreference resolution by bringing more context.

8.2 Background

8.2.1 In-document Coreference Resolution

In-document coreference resolution attempts to resolve coreference chains between entity mentions in a single document. Prior work has long utilized machine learning techniques, and can be mainly summarized as using the following general architectures:

- Pairwise mention scoring: In these work, the model learns a function $F(s, s')$ that measures how likely mention s corefers with mention s'. Usually, for each span x, the most likely antecedent $\arg\max_y F(x, y)$ is selected as the antecedent of x.

  In some work, the function F is learned as a *pairwise classifier* that outputs whether the two spans corefer. This thread of work includes Ng and Cardie (2002) and Bengtson and Roth (2008), both being driven by hand-crafted feature vectors.

  Other work approach the problem as a *ranking* problem, where the precedenting mentions of a specific mention is considered as candidate antecedents. The model then learns the rank these candidates with the relevance function F so that coreferring antecedents result on the top. These include Durrett and Klein (2013), Wiseman, Rush, Shieber, and Weston (2015), and K. Clark and Manning (2016), and later with Lee, L. He, Lewis, et al. (2017) and Lee, L. He, and Zettlemoyer (2018) that incorporates mention detection as a joint end-to-end neural model. Our discussion in this chapter is based on span-ranking methods, since it is relatively simple for neural models to learn (lacking complicated structure prediction), and also leading to state-of-the-art performance by incorporating recent large-scale language model pretraining methods, e.g. SpanBERT (Joshi, D. Chen, et al., 2020).

- Entity-level models allows coreference decisions to condition on entities (clusters of coreferring mentions) instead of on a mention level. These systems build up coreference chains with agglomerative clustering, starting from singleton clusters for each mention. These include Haghighi and Klein (2010), Stoyanov and Eisner (2012), K. Clark and Manning (2015), and Wiseman, Rush, and Shieber (2016).

- Latent-tree models approach the problem of coreference resolution as structured inference of a latent tree with span as nodes and coreference relation as edges. These include Fernandes, dos Santos, and Milidiú (2012), Martschat and Strube (2015), and Björkelund and Kuhn (2014).

8.2.2 Cross-document Coreference Resolution

Contrary to the traditional in-document coreference resolution task, *cross-document* coreference resolution's goal is to cluster all mentions of the same entity across different documents in a corpus. It is commonly modeled as a clustering problem (Bagga and Baldwin, 1998; Mann and Yarowsky, 2003; Gooi and Allan, 2004; Mayfield et al., 2009), where dependence for every pair of mentions in the document are created and classified for whether they are coreferent. These algorithm thus requires $\mathcal{O}(n^2)$ complexity over a corpus with n entity mentions, and operates in a *batch* mode where all pairwise relations are simultaneously considered.

D. Rao, McNamee, and Dredze (2010) cast the problem in a *streaming* fashion, where documents are processed one at a time and only a single time. This reduces the complexity to $\mathcal{O}(n)$ to achieve scalability across large document sets.

A most relevant work in our exploration in this chapter, and that I have collaborated on, is Sankepally et al. (2018), where the streaming idea is taking to a more extreme case, in that only a single mention is considered. Sankepally et al. (2018) is only concerned with mentions coreferring to a single query mention, hence casting the problem as a *coreferent mention retrieval* (CMR) problem. This falls into the theme of this thesis, where the problem is now retrieving candidate mentions in a corpus that corefers with the given query mention. Sankepally et al. (2018) utilized the discriminative IR proposed in Chapter 5 (T. Chen and Van Durme, 2017): here in this chapter we instantiate such a component with neural, dense vector representations instead of featurized sparse vectors.

8.3 In-document Model

We first review the state-of-the-art in-document coreference resolution model in Joshi, D. Chen, et al. (2020), which is based upon the higher-order model in Lee, L. He, and Zettlemoyer (2018), which, in turn, is based upon the span-ranking model of Lee, L. He, Lewis, et al. (2017).

8.3.1 Problem Formulation

In-document coreference resolution can be formulated as a set of antecedent assignments for every span i in a document $D = (w_1, \cdots, w_n)$. The set of all possible spans is $S = \{(l : r) \mid 1 \le l \le r \le n, r \le l + m\}$, where $(l : r)$ is a span (left and right inclusive), and m is the maximum number of tokens in a span. An order is assumed on the spans: first order by the left index, then the right index.

The coreference resolution task is therefore the assignment of each span $s_i \in S$ with an antecedent. For span s_i, the set of labels is $\mathcal{Y}_i = \{\varepsilon\} \cup \{1, \cdots, i-1\}$:

- ε is the dummy antecedent: it may cover the two possible scenarios: **(1)** s_i is not an entity mention; or **(2)** the span is an entity mention, but it is not coreferent with any previous span in S;
- If $1 \le j < i$ is the antecedent, it means that span s_j is a coreferent antecedent of span s_i.

Hence an in-document coreferent resolution model learns a mapping $s_i \mapsto \mathcal{Y}_i$. The set of decisions implies a final clustering of entity mentions, which can be recovered by taking the transitive closure of spans that are connected by antecedent predictions.

8.3.2 Mention Representations

Under a neural setup, a span should have a fixed-size vector representation to help build the overall model. The encoding of the text in the document has evolved with the progress of pretraining in NLP: from bidirectional LSTMs utilized in Lee, L. He, Lewis, et al. (2017) and Lee, L. He, and Zettlemoyer (2018), to using BERT (Joshi, Levy, et al., 2019), to the most recent SpanBERT (Joshi, D. Chen, et al., 2020), scores on various tasks continue to improve. In this section we will use the current state-of-the-art contextualized encoding SpanBERT (Joshi, D. Chen, et al., 2020) as our encoder.

We denote the vector representation of token w_i as $\mathbf{w}_i \in \mathbb{R}^d$.[1] To get a fixed length vector $\mathbf{s}_i$ for span s_i, the embedding of the left boundary token, the right boundary token, and a summarized vector of the span (computed by attention with a global query vector $\mathbf{a}$) is concatenated.

$$
\begin{aligned}
a_k &= \mathbf{a}^{\mathrm{T}} \mathbf{w}_k \\
p_{ik} &= \frac{\exp a_k}{\sum_{k=l_i}^{r_i} \exp a_k} \\
\mathbf{s}_i &= \left[\mathbf{w}_{l_i} \; ; \; \mathbf{w}_{r_i} \; ; \; \sum_{k=l_i}^{r_i} p_{ik} \mathbf{w}_k \right]
\end{aligned} \tag{8.1}
$$

Note that $\mathbf{s}_i \in \mathbb{R}^{3d}$.

We create a mention score function $F_{\mathrm{m}} : \mathbb{R}^{3d} \to \mathbb{R}$ to determine whether a span s_i is likely to be an entity mention. Note that our goal is recall here: we do not want to have valid entity mentions ignored by the model. F_{m} is modeled as a feed-forward neural network, and spans with higher scores are considered more likely to be entity mentions.

[1] For tokens tokenized into subwords units, its represetation is the average of the representations of its subwords.

The model first uses F_{m} to prune the number of spans to consider to effectively reduce the computational load of the model. According to Lee, L. He, and Zettlemoyer (2018), the number of spans to take into account is set to be $0.4\times$ the number of tokens in document D: it achieves a mention recall of around 92% as reported by Lee, L. He, and Zettlemoyer (2018) with BiLSTMs, but under our experiments this number is about 98% with SpanBERT.

8.3.3 Span Ranking

The crucial part of the model is to have a relevance function $F(\mathbf{s}_i, \mathbf{s}_j)$ that ranks candidate antecedents $\mathcal{Y}_i = \{\varepsilon\} \cup \{s_j\}_{1\le j<i}$ under the semantic relation of how likely s_j corefers with s_i. Under the theme of this thesis, we have a ranking problem here, where s_i is the query, $\mathcal{Y}_i$ is the mention candidate set, and F is the relevance function.

Lee, L. He, and Zettlemoyer (2018) utilized a method they termed "coarse-to-fine inference" to reduce the computation cost. It is essentially operating under the triage-then-rerank paradigm we discussed in Chapter 2. The method will be elaborated using ranking terminologies below.

Two relevance functions are defined: a "fine" relevance function that is accurate and heavyweight, and a "coarse" one that is relatively efficient to compute, and approximates the former:

- Approximate relevance function:

$$\tilde{F}(\mathbf{s}_i, \mathbf{s}_j) = F_\mathrm{m}(\mathbf{s}_i) + F_\mathrm{m}(\mathbf{s}_j) + F_\mathrm{c}(\mathbf{s}_i, \mathbf{s}_j) \tag{8.2}$$

  where F_m determines whether span j is likely to be an entity mention to be considered; and F_c is a pairwise function that is easy to compute. It takes the bilinear form:

$$F_\mathrm{c}(\mathbf{s}_i, \mathbf{s}_j) = \mathbf{s}_i^\mathrm{T} \mathbf{W}_\mathrm{c} \mathbf{s}_j \ , \tag{8.3}$$

  where $\mathbf{W}_\mathrm{c} \in \mathbb{R}^{3d \times 3d}$ is a weight matrix to be learned.

- Accurate relevance function:

$$F(\mathbf{s}_i, \mathbf{s}_j) = \tilde{F}(\mathbf{s}_i, \mathbf{s}_j) + F_\mathrm{a}(\mathbf{s}_i, \mathbf{s}_j) \ , \tag{8.4}$$

  where F_a is a concatenation-based model that is a feed-forward neural network stacked upon the concatenation of two span representations and their elementwise product:

$$F_\mathrm{a}(\mathbf{s}_i, \mathbf{s}_j) = \mathrm{FFNN}_\mathrm{a}([\mathbf{s}_i \ ; \ \mathbf{s}_j \ ; \ \mathbf{s}_i \odot \mathbf{s}_j \ ; \ \phi(\mathbf{s}_i, \mathbf{s}_j)]) \ , \tag{8.5}$$

  where ϕ is a manually constructed feature vector that encodes the distance between two spans.

Function F_a is the residual of F_1 except $\tilde{F}$: we force the concatenative model F_a to learn relations that the approximate model $\tilde{F}$ is not able to learn.

Under this setup, we could easily arrive at a cascading triage-then-rerank approach to coreference resolution:

- **(0) Enumerate all spans**: All spans whose length is less than m is considered:

$$S_0 = \{(l : r) \mid 1 \leq l \leq r \leq n, r \leq l + m\} \; ; \tag{8.6}$$

- **(1) Prune spans:** Keep the top K_1 spans based on the mention score $F_{\mathrm{m}}(\cdot)$:

$$S_1 = \underset{s \in S_0}{\arg\mathrm{top}_{K_1}} \; F_{\mathrm{m}}(s) \subseteq S_0 \; ; \tag{8.7}$$

- **(2) Triaging:** For each mention $s_i \in S_1$, get the top K_2 spans base on the approximate relevance function $\tilde{F}(s_i, \cdot)$:

$$S_2(s_i) = \underset{s_j \in S_1}{\arg\mathrm{top}_{K_2}} \; \tilde{F}(\mathbf{s}_i, \mathbf{s}_j) \subseteq S_1 \; ; \tag{8.8}$$

- **(3) Reranking:** Rerank the candidate mentions in $S_2(s_i)$ plus the dummy antecedent ε under the accurate relevance function $F(s_i, \cdot)$ and get the top one as the most likely antecedent:

$$\hat{y}_i = \underset{s_j \in S_2(s_i) \cup \{\varepsilon\}}{\arg\max} \; F(\mathbf{s}_i, \mathbf{s}_j) \in \mathcal{Y}_i \; . \tag{8.9}$$

The candidate set is reduced in every step: from $S_0 \to S_1 \to S_2(s_i) \to \{\hat{y}_i\}$, the size of these set goes from $|S_0| \to K_1 \to K_2 \to 1$. This coarse-to-fine process is the key to make the model perform reasonably fast but still performant.

8.3.4 High-order Inference

The model elaborated in the previous section suffers from a problem called the *consistency error*: There could be predictions that are locally consistent by not globally inconsistent. Take the following example from Wiseman, Rush, and Shieber (2016):

> SPEAKER 1: Um and [**I**]$_1$ think that is what's – Go ahead Linda.
>
> SPEAKER 2: Well and uh thanks goes to [**you**]$_2$ and to the media to help us... So our hat is off to [**all of you**]$_3$ as well.

Span pairs (I, you) and (you, all of you) are locally consistent since the plurality of "you" is not specified, but the span cluster (I, you, all of you) is globally inconsistent: the full cluster has mixed plurality. To alleviate this kind of problem that requires cluster-level information instead of just mention information, models should softly consider multiple hops in the predicted clusters. We will first look at the higher-order coreference resolution model in Lee, L. He, and Zettlemoyer (2018) designed to solve this problem.

The inference procedures takes N iterations. At each iteration, each the representation of each span s_i is updated with an attention mechanism that averages over previous representations weighted according to how likely each mention is to be an antecedent for span s_i. In essence, each span attends to other coreferent spans to update its representation.

The span representation in the previous section is the base case: the vector representation $\mathbf{s}_i$ is denoted as $\mathbf{s}_i^{(0)}$ to signify that it is the 0-th iteration.

At each iteration $0 \leq t < N$, we first compute the relevance scores for each candidate spans:

$$r_{ij}^{(t)} = F(\mathbf{s}_i^{(t)}, \mathbf{s}_j^{(t)}) \; . \tag{8.10}$$

Then an aggregated vector of its candidate antecedents are computed with r_{ij} as attention scores. This is similar to the context vector commonly used in machine translation decoders:

$$p_{ij}^{(t)} = \frac{\exp r_{ij}^{(t)}}{\sum\limits_{j \in \mathcal{Y}_i} \exp r_{ij}^{(t)}} \tag{8.11}$$

$$\mathbf{c}_i^{(t)} = \sum_{j \in \mathcal{Y}_i} p_{ij}^{(t)} \mathbf{s}_j^{(t)} \tag{8.12}$$

Finally, the current span representation $\mathbf{s}_i^{(t-1)}$ is updated by interpolation of itself with the aggregated vector, with a learned gating function:

$$\mathbf{g}_i^{(t)} = \sigma(\mathbf{W}_\mathrm{g} \cdot [\mathbf{s}_i^{(t)} \; ; \; \mathbf{c}_i^{(t)}]) \tag{8.13}$$

$$\mathbf{s}_i^{(t+1)} = \mathbf{g}_i^{(t)} \odot \mathbf{s}_i^{(t)} + (1 - \mathbf{g}_i^{(t)}) \odot \mathbf{c}_i^{(t)} \tag{8.14}$$

According to Lee, L. He, and Zettlemoyer (2018), 2nd order inference significantly outperforms first order inference, but orders higher than 2 do not bring significant additional benefit.

8.3.5 Training

In Lee, L. He, and Zettlemoyer (2018), they proposed to use a marginal cross-entropy loss to train the model. Specifically, since a span can have multiple correct antecedent spans, they assume that there is one gold antecedent span in this set. The probability of the "gold" antecedent span is thus marginalized across all these spans.

We can forgo the assumption of the existence of a single gold antecedent span here, and propose to use pairwise learning to rank methods here.

Observe that the accurate relevance function $F(s, s')$ can be seen as a ranking function that ranks candidate antecedents. All coreferring antecedents are considered as correct, hence are relevant candidates. The dummy ε antecedent is positive if span s has no coreferring antecedent, but negative otherwise. All other non-coreferring antecedent spans are clearly irrelevant samples.

To apply the approach as ranking, we consider all coreferring antecedents as relevant mentions and all non-coreferring mentions as irrelevant. The dummy ε serves as the threshold between the coreferring and the non-coreferring mentions. This idea is similar to what we propose in the previous chapter, where we use the parent node as the threshold between positive subtypes and negative subtypes. In essence, the model learns the following ranking relation:

$$s \succ \varepsilon \succ s', \quad s \in \mathcal{Y}, s' \notin \mathcal{Y} \tag{8.15}$$

Note that if there is no coreferring antecedent, the dummy node ε automatically became the highest ranking candidate in the candidate span set. This can be learned with a pairwise ranking loss similar to what we proposed in the previous chapter:

$$L_{s \succ \varepsilon} = [\alpha - F(x, y) + F(x, \varepsilon)]_+ \tag{8.16}$$

$$J_{\varepsilon \succ s'} = \sum_{y' \notin \mathcal{Y}} [(1 - \alpha) - F(x, \varepsilon) + F(x, s')]_+ \tag{8.17}$$

$$J_{s \succ s'} = \sum_{y' \notin \mathcal{Y}} [1 - F(x, s) + F(s, y')]_+ \tag{8.18}$$

$$J(x, \mathcal{Y}) = J_{s \succ \varepsilon} + J_{\varepsilon \succ s'} + J_{s \succ s'} \tag{8.19}$$

Again, α controls the precision/recall tradeoff: a larger α shifts the decision boundary towards the negative spans, thereby increasing recall; vice versa, a smaller α makes the model more strict in predicting coreferring spans, increasing precision. The margin hyperparameter can just be 1 since the last layer is linear—the scale can be learned into the weights of the last layer as a linear SVM.

8.4 Retrieval

The 2nd order inference problem is can be considered as an attention mechanism to other previous spans in text. Can we extend this to an attention to other spans, including spans in other documents rather than in the same document? The research question here is: "can we extrapolate the in-document coreference relevance function to cross-document scenarios?" We develop an approximate attention mechanism that attends to mentions in *all other documents in the corpus*. That is, instead of the spans $S(D)$ in document D, we are concerned with $\mathcal{S} = \bigcup_{D \in \mathcal{D}} S(D)$, the set of all entity mentions in the corpus $\mathcal{D}$.

As we have discussed in Chapter 2, for a retriever to run efficiently, there are 3 issues to be concerned with: query representation, candidate representation, and a relevance function that allows for fast nearest neigbor or maximum inner product search.

Our query and candidates here are clearly spans (that are deemed as entity mentions by the model) $s \in \mathcal{S}$, and they do have a natural dense-vector based representations $\mathbf{s} \in \mathbb{R}^{3d}$ here. Can we directly use these embeddings and retrieve under the accurate relevance function F as in Equation 8.9? The answer is no: The part F_a based on concatenated vectors cannot be computed efficiently if over all mentions in the corpus.

If the accurate relevant function F is infeasible here, can we use the approximate relevance function $\tilde{F}$? It turns that we can, but that requires some manipulations on the vector representations.

Recall the approximate relevance function $\tilde{F}$ between spans as described in Equation 8.2:

$$\tilde{F}(\mathbf{s}_i, \mathbf{s}_j) = F_{\mathrm{m}}(\mathbf{s}_i) + F_{\mathrm{m}}(\mathbf{s}_j) + \mathbf{s}_i^{\mathrm{T}} \mathbf{W}_{\mathrm{c}} \mathbf{s}_j \tag{8.20}$$

At the first glance, this does not seem readily reducible to a familiar form (i.e., either nearest neighbor search under a metric, or maximum inner product search). However, this is reducible to maximum inner product search under asymmetric transformations (similar to the trick employed in Neyshabur and Srebro (2015)) on both query and candidate vectors.

Note that

$$\tilde{F}(\mathbf{s}_i, \mathbf{s}_j) = \begin{bmatrix} \mathbf{s}_i \\ F_{\mathrm{m}}(\mathbf{s}_i) \\ 1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{W}_{\mathrm{c}} & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}_j \\ 1 \\ F_{\mathrm{m}}(\mathbf{s}_j) \end{bmatrix} . \tag{8.21}$$

We see that this relevance function is indeed decomposible, namely, it can be rewritten as the inner product of two vectors.

We define the following transformation functions on both query and candidate vectors:

$$\mathbf{t}_{\mathrm{Q}}(\mathbf{s}) = (\mathbf{W}_{\mathrm{c}}\mathbf{s} \; ; \; F_{\mathrm{m}}(\mathbf{s}) \; ; \; 1) \in \mathbb{R}^{3d+2} \; ; \tag{8.22}$$

$$\mathbf{t}_{\mathrm{C}}(\mathbf{s}) = (\mathbf{s} \; ; \; 1 \; ; \; F_{\mathrm{m}}(\mathbf{s})) \in \mathbb{R}^{3d+2} \; ; \tag{8.23}$$

Therefore,

$$\tilde{F}(\mathbf{s}_i, \mathbf{s}_j) = \mathbf{t}_{\mathrm{Q}}(\mathbf{s}_i) \cdot \mathbf{t}_{\mathrm{C}}(\mathbf{s}_j) \tag{8.24}$$

This shows that the approximate relevance function $\tilde{F}$ is indeed decomposable into the inner product of two vectors, but these vectors require some augmentation. This augmentation technique adds 2 extra dimensions to the vector, and contains the score computed in F_m.

This decomposition shares the spirit of discriminative retrieval as we discussed in Chapter 5 and Chapter 6: The model learns a ranker and reuses its parameters to inform the retriever what to retrieve. The retriever first maps a query to a representation (here $\mathbf{t}_\mathrm{Q}$) so that its most relevant candidates can be retrieved using MIPS.

Now we can perform retrieval of coreferent mentions across documents. First index all the candidate vector representations $\mathbf{t}_\mathrm{C}(\mathbf{s}')$ of spans $s' \in \mathcal{S}$. At inference time, for each query span s, we retrieve the top-$\bar{K}$ spans across the whole corpus, denoted as $\bar{S}(s)$:

$$\bar{S}(s) = \underset{s' \in \mathcal{S}}{\arg\,\mathrm{top}_{K_C}}\ \tilde{F}(\mathbf{s}, \mathbf{s}') = \underset{s' \in \mathcal{S}}{\arg\,\mathrm{top}_{K_C}}\ \mathbf{t}_\mathrm{Q}(\mathbf{s}) \cdot \mathbf{t}_\mathrm{C}(\mathbf{s}') \subseteq \mathcal{S} \tag{8.25}$$

Then we compute the attention mechanism in $\bar{S}(s)$ instead of the whole set $\mathcal{S}$. This is a good approximation of the partition function since the relevance scores follow a long-tail distribution (see the diagnostic section below for validation):

$$\sum_{s' \in \bar{S}(s)} \exp \tilde{F}(\mathbf{s}, \mathbf{s}') \approx \sum_{s' \in \mathcal{S}} \exp \tilde{F}(\mathbf{s}, \mathbf{s}') \tag{8.26}$$

Now we can proceed with a second-order inference procedure that not only attends to coreferring mentions in the same document, but across a whole corpus:

$$\bar{r}_{s'} = \tilde{F}(\mathbf{s}, \mathbf{s}') \quad s' \in \bar{S}(s) \tag{8.27}$$

$$\bar{p}_{s'} = \frac{\exp r_{s'}}{\sum_{s' \in \bar{S}(s)} \exp r_{s'}} \tag{8.28}$$

$$\bar{\mathbf{c}} = \sum_{s' \in \bar{S}(s)} p_{s'} \mathbf{s}' \tag{8.29}$$

Vector $\bar{\mathbf{c}}$ is an weighted aggregation of all spans in the whole corpus that is relevant to the current span. The 2nd order inference algorithm update the representation of a span with not only the in-document context $\mathbf{c}$, but also with the cross-document corpus-wide context $\bar{\mathbf{c}}$. The gating function in Equation 8.13 is also modified to account for a third component, the cross-document context:

$$[\mathbf{g}_s \, ; \, \mathbf{g}_c \, ; \, \mathbf{g}_{\bar{c}}] = \operatorname{softmax}(\mathbf{W}_{\mathrm{g}} \cdot [\mathbf{s}_i^{(t)} \, ; \, \mathbf{c}_i^{(t)} \, ; \, \bar{\mathbf{c}}_i^{(t)}]) \in \mathbb{R}^{3d \times 3} \tag{8.30}$$

$$\mathbf{s}_i^{(t+1)} = \mathbf{g}_s \odot \mathbf{s}_i^{(t)} + \mathbf{g}_c \odot \mathbf{c}_i^{(t)} + \mathbf{g}_{\bar{c}} \odot \bar{\mathbf{c}}_i^{(t)} \tag{8.31}$$

Relation to Shifted Inner Product Similarities Okuno, Kim, and Shimodaira (2019) proved that any similarity function $F : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ over element set $\mathcal{X}$ in the form of $F(x, y) = \mathbf{f}(x) \cdot \mathbf{f}(y) + g(x) + g(y)$ (termed a *shifted* inner product similarity, where $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{X} \to \mathbb{R}$ are both parameterized neural functions) can approximate any conditional positive-definite (CPD) kernel (Schölkopf, 2000), whereas a normal inner product similarity in the form of $F(x, y) = \mathbf{f}(x) \cdot \mathbf{f}(y)$ can only approximate any positive-definite kernel (the normal kernels utilized in SVMs). The approximate relevance function $\tilde{F}$ between spans here is an instance of shifted inner product similarity.

CPD kernels are known to be inherently more expressive than normal kernels, and our derivation here provides a method for reducing maximum *shifted* inner product search to a maximum inner product search problem with asymmetric tricks (Equations (8.22) to (8.24)).

8.5 Experiments and Discussions

We conduct our experiments based on the higher-order neural coreference model (Lee, L. He, and Zettlemoyer, 2018), with the state-of-the-art pretrained encoder SpanBERT (Joshi, D. Chen, et al., 2020) for the coreference task. Our implementation is based on the neural coreference model in the `allennlp-models` package,[2] that contains a set of state-of-the-art NLP models based upon AllenNLP (Gardner et al., 2018).

Dataset We use the English portion of the coreference resolution dataset from the CoNLL-2012 shared task (Pradhan et al., 2012), which is in turn a subset of the OntoNotes 5.0 dataset (Weischedel, Palmer, et al., 2013). The dataset is diverse in different text genres, and contains 2,802 documents in the training set, 343 in dev, and 348 in test.

[2] `https://github.com/allenai/allennlp-models/tree/master/allennlp_models/coref`.

General Setup We use the SPANBERT-LARGE variant of SpanBERT, where the encoding dimensionality for each token is 1,024. We use the coreference-fine-tuned SpanBERT weights published by Joshi, D. Chen, et al. (2020), and freeze these weights to accelerate training and to reduce the GPU memory footprint.

We inherit most of the hyperparameters from the AllenNLP code: the maximum number of tokens per span is 30; the maximum number of spans to retain in the mention filtering stage is 0.4 per token, the number of spans selected in the triaging phase $K_2 = 50$, and the dropout probability for the mention representation is $p_{\mathrm{D}} = 0.3$.

Metrics To evaluate in-document coreference resolution, we follow the metrics defined in prior work by using the official evaluation metrics in the CoNLL-2012 shared task. These include the precision, recall, and F_1 scores of MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and $\mathrm{CEAF}_{\phi 4}$ (Luo, 2005). The main evaluation metric is the average F_1 score of the three metrics.

8.5.1 Ranking Loss in In-document Model

We investigate how the ranking-based loss performs for the in-document coreference resolution task, as compared to the marginalized cross-entropy utilized in Lee, L. He, and Zettlemoyer (2018). The main hyperparameter to tune for this experiment is α: the ratio of the margin between coreferring antecedents/dummy and dummy/non-coreferring spans. Our best result is reported in Table 8.1. The best-performing model has $\alpha = 0.15$.

Model	MUC			B^3			$\text{CEAF}_{\phi 4}$			Avg.
	P	R	F_1	P	R	F_1	P	R	F_1	F_1
Joshi, D. Chen, et al. (2020)	**85.8**	84.8	**85.3**	78.3	77.9	78.1	76.4	**74.2**	**75.3**	**79.6**
Ranking-based loss	83.7	**85.0**	84.3	**78.5**	**81.0**	**79.8**	**79.5**	69.8	74.4	79.5

Table 8.1: Results of ranking loss vs. marginalized cross entropy loss for in-document coreference resolution.

The final result is not much different: we observed a 0.1% decrease of performance in the final average F_1 score, indicating that a ranking-based loss, when properly trained, is similar to a marginalized cross-entropy loss.[3] However, on the metric B^3, the ranking-based loss is superior, especially in recall, but much worse on $\text{CEAF}_{\phi 4}$. This is interesting since the recall of B^3 is the proportion of its actual coreferents that the system thinks are coreferent with it, averaged across all mentions,[4] and the CEAF metric computes the performance on the entity level instead of on the mention level as in B^3 (Cai and Strube, 2010). Our gain in B^3 recall but loss in $\text{CEAF}_{\phi 4}$ recall indicate that a ranking-based loss performs better on the mention level but worse on the entity level.

8.5.2 Corpus-wide Second-order Inference

We experiment with incorporating corpus-wide second-order inference, i.e., attending over the entire corpus, to see if it improves the performance of in-document coreference resolution. The main hyperparameter to tune here is the number of spans to retrieve $\bar{K}$

[3] This is corrobrated in Lee, L. He, Lewis, et al. (2017) where they considered a ranking-based loss but has not reported the result.

[4] See Brendan O'Connor's blogpost *Probabilistic interpretation of the B3 coreference resolution metric* that discusses these metrics at `https://brenocon.com/blog/2013/08/probabilistic-interpretation-of-the-b3-coreference-resolution-metric/`.

Model	MUC			B^3			CEAF$_{\phi 4}$			Avg.
	P	R	F_1	P	R	F_1	P	R	F_1	F_1
$\bar{K}$ = 0 (w/o external attention)	85.8	**84.8**	**85.3**	78.3	77.9	78.1	76.4	**74.2**	**75.3**	79.6
$\bar{K}$ = 2	**86.9**	82.6	84.7	**82.8**	77.8	**80.3**	81.2	68.6	74.4	79.7
$\bar{K}$ = 4	86.7	82.6	84.6	82.4	77.9	80.2	**81.3**	68.4	74.3	79.7
$\bar{K}$ = 8	86.5	83.1	84.7	82.1	**78.4**	80.2	**81.3**	68.7	74.5	79.8
$\bar{K}$ = 16	86.7	82.8	84.7	82.5	78.1	**80.3**	**81.3**	68.6	74.4	79.8
$\bar{K}$ = 32	**86.9**	82.8	84.8	82.7	78.0	**80.3**	**81.3**	68.7	74.5	**79.9**

Table 8.2: Results of corpus-wide second-order inference for in-document coreference resolution.

from the entire corpus: we select the number from $\{2, 4, 8, 16, 32\}$: the more to retrieve, the slower the training process would be.

We index all the mentions found in the training and dev set respectively (first pass decoding), then at training time we retrieve from the training mention index, whereas at validation time we retrieve from the dev mention index. This can be seen as a kind of *two pass decoding*, where first the model detects all possible mentions for a document set, index these mentions, then the model predicts coreference chains with corpus-wide second-order inference conditioned on all these mentions discovered in the dataset. In the CoNLL-2012 dataset here, the training set has 319,426 mentions to retrieve from, the dev set 38,581, and the test set 42,100.

Results are shown in Table 8.2. It shows that attending over the corpus brings very modest improvement to the overall performance. Again, the improvement over B^3 is more significant, but not much overall.

8.5.3 Diagnostics and Discussions

We perform some diagnostics on our proposed approach, namely, we investigate how well the approximated second-order inference under cross-document coreference works.

Retrieval for approximating the partition function In Equation 8.26, we proposed that the sum of the retrieved elements can approximate the partition function $Z(s) = \sum_{s' \in \mathcal{S}} \exp \tilde{F}(\mathbf{s}, \mathbf{s}')$ over all mention spans in the corpus based on the hypothesis of the long-tail distribution. Here we randomly sample 100 mention spans from the dev set of the CoNLL-2012 dataset, and plot the scores $\tilde{F}(\mathbf{s}, \mathbf{s}')$ to see whether it follows a long-tail distribution.

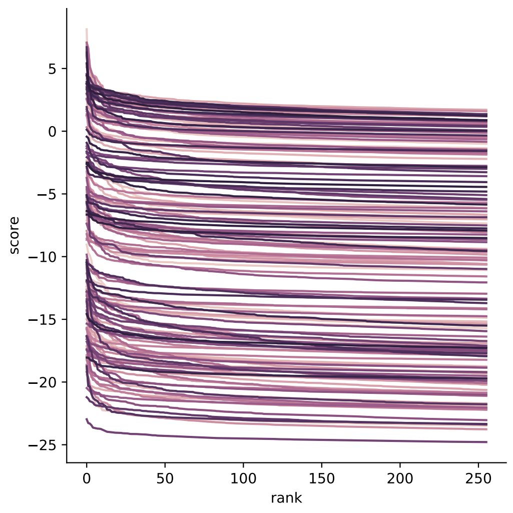


Figure 8.1: Raw relevance scores of the retrieved candidate mentions, with each line in the plot representing one query mention span.

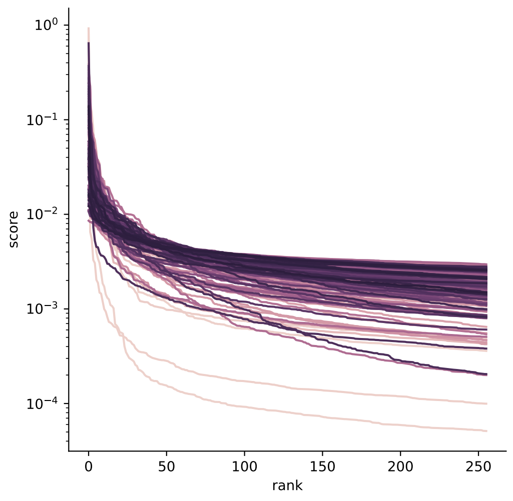


Figure 8.2: Softmax-normalized relevance scores of the retrieved candidate mentions, on a log scale for the y-axis.

Figure 8.1 plots the raw scores of the retrieved candidate mentions for each sampled query mention span, with x-axis as the rank (here we did top-k search with $k = 256$), and

Figure 8.2 plots the softmax-normalized scores used in the second-order inference procedure. We clearly observe the long-tail distribution of these scores here, hence justifying our approximation of the partition function.

Proportion of in-document mentions We investigate the mentions retrieved for each mention that has at least one other coreferent mention in the same document, and found out that most top mentions retrieved are in-document, and out-of-document retrieved mentions are pretty random and most often does not corefer with the query mention.

K mentions to be retrieved	2	4	6	16	32
Proportion of retrieved mentions to be in-doc	59.0	39.2	28.6	19.8	14.0

Table 8.3: Proportion of retrieved mentions that are in the same document as the query mention.

In Table 8.3, the randomly-sampled mention set in the previous diagnostic test is reused, while filtering out singleton mentions. The proportion of retrieved mentions that are in the same document is very high at $K = 2$ or 4: this shows that in-document coreferring mentions are far more likely to be retrieved that cross-document mentions.

Cross-document retrieved mentions are also *not* coreferring to the query mention: in our sampled subset of test spans, if we take $K = 16$ and manually inspect the retrieved cross-document mentions (note that there is no annotations for cross-document coreference in the CoNLL-2012 dataset), we found out that only about 4% are correct: many shares topical context but are not coreferent. For example, a query span "*the city*" (referring to the city of

Hong Kong in the document) retrieves mentions such as "*Hollywood of the East*" (referring to the film industry of Hong Kong in the containing document, but not the city itself), "*the Hong Kong government*", "*Shenzhen*" (an adjacent city just across a river, but in mainland China), "*Taiwan*" (a close geo-political entity), or "*Hong Kong Wetland Park*".

This shows that the in-document trained coreference relevance function does not readily extrapolate to cross-document scenarios because the encodings learned for the mentions are highly contextualized, and the performance gains ($\approx$ 0.3% gain in average F_1) may come from the information of later coreferent spans (the base model of Lee, L. He, and Zettlemoyer (2018) does not consider spans *after* the current span). For future work, one might restrict the search to only in-document mentions or only cross-document mentions. Additionally, one might try to fine-tune the relevance function to let it learn cross-document coreference links by using cross-document coreference datasets such as the one in Sankepally et al. (2018).

Part IV

Conclusions

Chapter 9

Conclusions

9.1 Contributions

This thesis has made various contributions to the problem of ranking and retrieval under a wide range of semantic relevance relations. In Chapter 3, we have shown that by extending the traditional categorical notion of natural language inference to *uncertain* natural language inference, models can learn to predict scalar subjective probabilities instead of categorical labels, making the predictions closer to human judgements. By treating the model as a relevance function under the semantic relation of inference, we devise regression and ranking methods to model such inference, and led to predictions that better correlate with humans. In Chapter 4, we consider the problem of situation frame detection, where we consider the relevance function of whether a document evokes a specific situation frame. We devised ranking methods based on annotators or HITs, illustrating modeling methods that could potentially ameliorate per-annotator or per-HIT bias in data elicitation. In Chapter 5, we consider the relevance function of a candidate passage answering a given natural language question, and devised a featurized sparse-vector based method that could efficiently and scalably retrieve candidate answer passages for a given question. This method significantly improves triaging recall that could benefit downstream reranking models. In Chapter 6, we apply the same idea, but based on neural, dense embeddings for question answering and similar questions retrieval. Although our results are not competitive, later work has shown that this research direction is viable with more advanced dense representations and better sampling strategies.

The contributions of Part II of this thesis center on semantic relevance over entity mentions rather than text snippets. Chapter 7 proposed methods to model the semantic relevance of a mention belonging to a specific entity type in the task of fine-grained entity typing. Under the tree-structured ontology of entity types, we devised ranking methods that fully embraces the tree structure of the ontology, leading to state-of-the-art results for the task. In Chapter 8, we viewed coreference resolution models as span-ranking models whose relevance function models whether two mention spans corefer. We explored extrapolating the relevance function learned in an in-document scenario to a cross-document scenario, and proposed methods that could retrieve coreferring mentions across multiple documents with neurally learned dense representations.

9.2 Future Work

The research presented in this thesis initiates further research questions. A main concern for all of these ranking and retrieval method is *representation learning*: methods for generating better representations that could facilitate more accurate retrieval are of utmost importance. With the rapid advancement of pretraining methods like ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), SpanBERT (Joshi, D. Chen, et al., 2020), etc., better representations can be generated. However, the geometry of the space of the representations remain to be studied, for example, the anisotropy of the generated vectors Ethayarajh (2019) may make approximate dense vector retrieval methods perform suboptimally.

Chapter 3 and Chapter 4 raises the question of annotation bias with respect to different annotators and HITs. More research into ranking methods to ameliorate these biases may be studied. The problem of natural language inference can be considered the inverse of *abductive reasoning* (Bhagavatula et al., 2020), where we can apply retrieval methods proposed in this thesis to perform corpus-wide abductive reasoning by finding evidences in corpus that supports certain claims.

The dense retrieval method for open-domain question answering in Chapter 6 can further faciliate multi-hop question answering, where systems iteratively retrieve paragraphs in a large corpus for multi-hop reasoning. A close work following this thread is Feldman and El-Yaniv (2019).

Chapter 7 addresses the problem of typing under a tree-structured ontology. However, types in an ontology may be structured as a directed acyclic graph (DAG), as DAGs extends trees by allowing multiple inheritance of types rather than single inheritance. Representation learning methods like box embeddings (Vilnis et al., 2018; X. Li et al., 2019) or hyperbolic embeddings in the Poincaré ball space (López, Heinzerling, and Strube, 2019) are possible ways to deal with DAG-structured ontologies. Extension of ranking methods to these spaces remain unsolved. Another thread of future research is the proper treatment of *nested* entities.

The fine-grained entity typing task may be further generalized to include *entity linking*: when the types gets finer and finer, they arrive at the level of singleton entities, effectively becoming entity linking. Future research can seek to unify entity typing and linking under the same task, and learn representations for both types and entities in the same space (Euclidean, or alternatives such as Poincaré). Since the number of these types and entities might be large (e.g. more than 5 million Wikipedia entities), typing and linking can be again cast as a *retrieval* problem, instigating further research in scalable retrieval for these spaces.

Finally, Chapter 8 raises the possibility of using document-level information for coreference resolution, by treating the entire corpus as a candidate set to retrieve from. While the work presented in that chapter is only exploratory, more could be done to refine the representations learned to execute cross-document coreference mention search, thus providing a neural dense-vector based solution to the coreferent mention retrieval (CMR) problem raised by Sankepally et al. (2018). Treating all mentions as a candidate set also opens research directions such as question answering or slot filling by directly retrieving from a set of mentions.

More broadly, future work may extend the ranking and retrieval paradigm discussed in this thesis to a wide range of semantic and information extraction tasks (e.g. entity linking, knowledge base question answering, etc.), and these ranking and retrieval modules can serve as components in larger AI systems such as dialogue agents, enabling these agents the capability to perform scalable inference with large-scale corpora or knowledge bases.

Appendix A

Implementation Details

This section elaborates software engineering details with respect to the retrieval services, for both sparse vector-based retrieval systems as described in Chapter 5, and dense vector-based retrieval systems as described in Chapter 6 and Chapter 8.

A retrieval system for queries in $\mathcal{Q}$ and candidates in $C \subseteq \mathcal{C}$ are implemented as two distinct online services, called `search` and `fetch` here, and two offline service `store` and `index` that stores the raw candidates and the computed representations respectively. We give each candidate a string-valued ID.

The service `search` executes maximum inner product search for sparse or dense vectors based on an vector index and a query, and returns top-k candidate IDs. The `fetch` service fetches candidate objects given their IDs.

These routines can be summarized in Table A.1.

A generic `search` algorithm can be described as below in Algorithm 3.

Routine	Signature
CANDIDATE-REPR ($\mathbf{f}_C$)	$\mathcal{C} \to \mathcal{R}_C$
QUERY-REPR ($\mathbf{f}_Q$)	$\mathcal{Q} \to \mathcal{R}_Q$
TRANSFORM ($\mathbf{t}$)	$\mathcal{R}_Q \to \mathcal{R}_C$
STORE	`Iterable[(str,`$\mathcal{C}$`)]` $\to$ `Store`
FETCH	`(Store, List[str])` $\to$ `List[`$\mathcal{C}$`]`
INDEX	`Iterable[(str,`$\mathcal{R}_C$`)]` $\to$ `Index`
SEARCH	`(Index,` $\mathcal{R}_C$`, int)` $\to$ `List[str]`

Table A.1: Common routines in retrieval services.

Algorithm 3 Generic retrieval algorithm

```
1: function RETRIEVE(Index, Store, $q \in \mathcal{Q}, k \in \mathbb{N}$)
2:     $\mathbf{q} \leftarrow$ QUERY-REPR($q$)                          ▷ Query representation
3:     $\tilde{\mathbf{q}} \leftarrow$ TRANSFORM($\mathbf{q}$)                        ▷ Transformed query
4:     $(i_1, \cdots, i_k) \leftarrow$ SEARCH(Index, $\tilde{\mathbf{q}}$, $k$)           ▷ Top-$k$ search
5:     $(c_1, \cdots, c_k) \leftarrow$ FETCH(Store, $(i_1, \cdots, i_k)$)     ▷ Fetch candidate objects
6: return $(c_1, \cdots, c_k)$
7: end function
```

Throughout the experiments in this thesis, STORE and FETCH are implemented using the Berkeley DB[1] key-value store. Each key (string ID) and value (candidate objects, can be a string or a multi-dimensional array when encoded) are serialized as byte arrays that can be stored in a Berkeley DB database. Multi-dimensional arrays are represented as NumPy (Harris et al., 2020) `np.ndarrays` then serialized using `msgpack-numpy`[2].

[1] `https://www.oracle.com/database/berkeley-db/`.
[2] `https://github.com/lebedov/msgpack-numpy`.

A.1 Retrieval with Sparse Vectors

For sparse vector-based retrieval systems, we use Apache Lucene[3] as the retrieval engine to implement the INDEX and SEARCH services. A sparse vector is converted to a stream of Lucene terms, with the weights encoded using the "payload" functionality in Lucene (that allows the addition of arbitrary bytes for each term stored). We extend the standard `org.apache.lucene.search.similarities.ClassicSimilarity` to account for the addition of payloads (that encode weights), so that our customized inner product search can be efficiently executed in Lucene. We release this library as `probe`[4].

A.2 Retrieval with Dense Vectors

For dense vector-based retrieval systems, the underlying engine for the INDEX and SEARCH services is FAISS[5] (Johnson, Douze, and Jégou, 2017). We use the `faiss.IndexIVFFlat` and `faiss.MultiIndexQuantizer` for a retrieval system based on the optimized product quantization (Ge et al., 2013) method.

[3] `https://lucene.apache.org/`.

[4] `https://github.com/ctongfei/probe`.

[5] `https://github.com/facebookresearch/faiss`.

Bibliography

Abhishek, Ashish Anand, and Amit Awekar (2017). “Fine-Grained Entity Type Classification by Jointly Learning Representations and Label Embeddings”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pp. 797–807. URL: https://www.aclweb.org/anthology/E17-1075/.

Amiri, Hadi, Philip Resnik, Jordan L. Boyd-Graber, and Hal Daumé III (2016). “Learning Text Pair Similarity with Context-sensitive Autoencoders”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. URL: https://doi.org/10.18653/v1/p16-1177.

Ammar, Waleed, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith (2016). “Massively Multilingual Word Embeddings”. In: *CoRR* abs/1602.01925. arXiv: 1602.01925. URL: http://arxiv.org/abs/1602.01925.

Andoni, Alexandr and Piotr Indyk (2006). “Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions”. In: *47th Annual IEEE Symposium on*

Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings, pp. 459–468. URL: https://doi.org/10.1109/FOCS.2006.49.

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre (2016). “Learning principled bilingual mappings of word embeddings while preserving monolingual invariance”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 2289–2294. URL: https://doi.org/10.18653/v1/d16-1250.

Bagga, Amit and Breck Baldwin (1998). “Entity-Based Cross-Document Coreferencing Using the Vector Space Model”. In: *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL ’98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference*, pp. 79–85. URL: https://www.aclweb.org/anthology/P98-1012/.

Bellman, R., Rand Corporation, and Karreman Mathematics Research Collection (1957). *Dynamic Programming*. Rand Corporation research study. Princeton University Press. URL: https://books.google.com/books?id=wdtoPwAACAAJ.

Bengtson, Eric and Dan Roth (2008). “Understanding the Value of Features for Coreference Resolution”. In: *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu,* *Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 294–303. URL: `https://www.aclweb.org/anthology/D08-1031/`.

Bentley, Jon Louis (1975). "Multidimensional Binary Search Trees Used for Associative Searching". In: *Commun. ACM* 18.9, pp. 509–517. URL: `http://doi.acm.org/10.1145/361002.361007`.

Bernardy, Jean-Philippe, Rasmus Blanck, Stergios Chatzikyriakidis, and Shalom Lappin (2018). "A compositional Bayesian semantics for natural language". In: *Proceedings of the First International Workshop on Language Cognition and Computational Models*, pp. 1–10.

Bhagavatula, Chandra, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen-tau Yih, and Yejin Choi (2020). "Abductive Commonsense Reasoning". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. URL: `https://openreview.net/forum?id=Byg1v1HKDB`.

Bilotti, Matthew W., Paul Ogilvie, Jamie Callan, and Eric Nyberg (2007). "Structured retrieval for question answering". In: *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pp. 351–358. URL: `https://doi.org/10.1145/1277741.1277802`.

Björkelund, Anders and Jonas Kuhn (2014). "Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features". In: *Proceedings* *of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pp. 47–57. URL: `https://doi.org/10.3115/v1/p14-1005`.

Bollacker, Kurt D., Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor (2008). “Freebase: a collaboratively created graph database for structuring human knowledge”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pp. 1247–1250. URL: `https://doi.org/10.1145/1376616.1376746`.

Bowman, Samuel R., Gabor Angeli, Christopher Potts, and Christopher D. Manning (2015). “A large annotated corpus for learning natural language inference”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 632–642. URL: `https://doi.org/10.18653/v1/d15-1075`.

Buckley, Chris and Ellen M. Voorhees (2004). “Retrieval evaluation with incomplete information”. In: *SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 25-29, 2004*. Ed. by Mark Sanderson, Kalervo Järvelin, James Allan, and Peter Bruza. ACM, pp. 25–32. URL: `https://doi.org/10.1145/1008992.1009000`.

Burges, Christopher J. C., Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender (2005). “Learning to rank using gradient descent”. In: *Machine Learning, Proceedings of the Twenty-Second International Con-*

ference (ICML 2005), Bonn, Germany, August 7-11, 2005, pp. 89–96. URL: https://doi.org/10.1145/1102351.1102363.

Cai, Jie and Michael Strube (2010). "Evaluation Metrics For End-to-End Coreference Resolution Systems". In: *Proceedings of the SIGDIAL 2010 Conference, The 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 24-15 September 2010, Tokyo, Japan*, pp. 28–36. URL: https://www.aclweb.org/anthology/W10-4305/.

Charikar, Moses (2002). "Similarity estimation techniques from rounding algorithms". In: *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pp. 380–388. URL: https://doi.org/10.1145/509907.509965.

Charniak, Eugene (1972). *Toward A Model Of Children"s Story Comprehension*. Tech. rep. AITR-266. Massachusetts Institute of Technology. URL: https://dspace.mit.edu/handle/1721.1/6892.

Chen, Danqi, Adam Fisch, Jason Weston, and Antoine Bordes (2017). "Reading Wikipedia to Answer Open-Domain Questions". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 1870–1879. URL: https://doi.org/10.18653/v1/P17-1171.

Chen, Tongfei, Yunmo Chen, and Benjamin Van Durme (2020). "Hierarchical Entity Typing via Multi-level Learning to Rank". In: *Proceedings of the 58th Annual Meeting* *of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 8465–8475. URL: https://www.aclweb.org/anthology/2020.acl-main.749/.

Chen, Tongfei, Zhengping Jiang, Adam Poliak, Keisuke Sakaguchi, and Benjamin Van Durme (2020). “Uncertain Natural Language Inference”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 8772–8779. URL: https://www.aclweb.org/anthology/2020.acl-main.774/.

Chen, Tongfei, Chetan Naik, Hua He, Pushpendre Rastogi, and Lambert Mathias (2019). “Improving Long Distance Slot Carryover in Spoken Dialogue Systems”. In: *Proceedings of the First Workshop on NLP for Conversational AI, Florence, Italy, July 28–August 2, 2019*, pp. 96–105. URL: https://www.aclweb.org/anthology/W19-4111.

Chen, Tongfei and Benjamin Van Durme (2017). “Discriminative Information Retrieval for Question Answering Sentence Selection”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pp. 719–725. URL: https://doi.org/10.18653/v1/e17-2114.

Choi, Eunsol, Omer Levy, Yejin Choi, and Luke Zettlemoyer (2018). “Ultra-Fine Entity Typing”. In: *Proceedings of the 56th Annual Meeting of the Association for Computa-*

tional Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, pp. 87–96. URL: https://www.aclweb.org/anthology/P18-1009/.

Clark, Kevin and Christopher D. Manning (2015). “Entity-Centric Coreference Resolution with Model Stacking”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pp. 1405–1415. URL: https://doi.org/10.3115/v1/p15-1136.

Clark, Kevin and Christopher D. Manning (2016). “Deep Reinforcement Learning for Mention-Ranking Coreference Models”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 2256–2262. URL: https://doi.org/10.18653/v1/d16-1245.

Cooper, Robin, Dick Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and Steve Pulman (1996). *Using the Framework*. Tech. rep. The FraCaS Consortium. URL: ftp://ftp.cogsci.ed.ac.uk/pub/FRACAS/del16.ps.gz.

Cooper, Robin, Simon Dobnik, Staffan Larsson, and Shalom Lappin (2015). “Probabilistic type theory and natural language semantics”. In: *Linguistic issues in language technology* 10.4, pp. 1–45.

Cooper, William S., Fredric C. Gey, and Daniel P. Dabney (1992). “Probabilistic Retrieval Based on Staged Logistic Regression”. In: *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992*, pp. 198–210. URL: https://doi.org/10.1145/133160.133199.

Dagan, Ido, Oren Glickman, and Bernardo Magnini (2005). “The PASCAL Recognising Textual Entailment Challenge”. In: *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, pp. 177–190. URL: https://doi.org/10.1007/11736790%5C_9.

Dai, Hongliang, Donghong Du, Xin Li, and Yangqiu Song (2019). “Improving Fine-grained Entity Typing with Entity Linking”. 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, Hong Kong, China, November 3-7, 2019*, pp. 6209–6214. URL: https://www.aclweb.org/anthology/D19-1643/.

Dasgupta, Sanjoy and Kaushik Sinha (2013). “Randomized partition trees for exact nearest neighbor search”. In: *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, pp. 317–337. URL: http://proceedings.mlr.press/v30/Dasgupta13.html.

Davidson, Donald (1967). “Truth and meaning”. In: *Philosophy, Language, and Artificial Intelligence*. Springer, pp. 93–111.

Del Corro, Luciano, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum (2015). “FINET: Context-Aware Fine-Grained Named Entity Typing”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 868–878. URL: `https://doi.org/10.18653/v1/d15-1103`.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). “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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. URL: `https://www.aclweb.org/anthology/N19-1423/`.

Duh, Kevin (2009). “Learning to rank with partially-labeled data”. PhD thesis.

Durrett, Greg and Dan Klein (2013). “Easy Victories and Uphill Battles in Coreference Resolution”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1971–1982. URL: `https://www.aclweb.org/anthology/D13-1203/`.

Ethayarajh, Kawin (2019). “How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings”. 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, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Association for Computational Linguistics, pp. 55–65. URL: `https://doi.org/10.18653/v1/D19-1006`.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin (2008). "LIBLINEAR: A Library for Large Linear Classification". In: *J. Mach. Learn. Res.* 9, pp. 1871–1874. URL: `https://www.jmlr.org/papers/volume9/fan08a/fan08a.pdf`.

Feldman, Yair and Ran El-Yaniv (2019). "Multi-Hop Paragraph Retrieval for Open-Domain Question Answering". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, pp. 2296–2309. URL: `https://doi.org/10.18653/v1/p19-1222`.

Fernandes, Eraldo R., Cícero Nogueira dos Santos, and Ruy Luiz Milidiú (2012). "Latent Structure Perceptron with Feature Induction for Unrestricted Coreference Resolution". In: *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning - Proceedings of the Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes, EMNLP-CoNLL 2012, July 13, 2012, Jeju Island, Korea*, pp. 41–48. URL: `https://www.aclweb.org/anthology/W12-4502/`.

Ferrucci, David A., Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty (2010). "Building Watson: An Overview of the DeepQA Project". In: *AI Magazine* 31.3, pp. 59–79. URL: https://doi.org/10.1609/aimag.v31i3.2303.

Gao, Luyu, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan (2020). "Complementing Lexical Retrieval with Semantic Residual Embedding". In: *CoRR* abs/2004.13969. arXiv: 2004.13969. URL: https://arxiv.org/abs/2004.13969.

Gardner, Matt, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer (July 2018). "AllenNLP: A Deep Semantic Natural Language Processing Platform". In: *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Melbourne, Australia: Association for Computational Linguistics, pp. 1–6. URL: https://www.aclweb.org/anthology/W18-2501.

Garrette, Dan, Katrin Erk, and Raymond J. Mooney (2011). "Integrating Logical Representations with Probabilistic Information using Markov Logic". In: *Proceedings of the Ninth International Conference on Computational Semantics, IWCS 2011, January 12-14, 2011, Oxford, UK*. URL: https://www.aclweb.org/anthology/W11-0112/.

Ge, Tiezheng, Kaiming He, Qifa Ke, and Jian Sun (2013). "Optimized Product Quantization for Approximate Nearest Neighbor Search". In: *2013 IEEE Conference on Com-*

puter Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013, pp. 2946–2953. URL: https://doi.org/10.1109/CVPR.2013.379.

Gey, Fredric C. (1994). “Inferring Probability of Relevance Using the Method of Logistic Regression”. In: *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pp. 222–231. URL: https://dl.acm.org/doi/abs/10.5555/188490.188560.

Giampiccolo, Danilo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, Elena Cabrio, and Bill Dolan (2008). “The Fourth PASCAL Recognizing Textual Entailment Challenge”. In: *Proceedings of the First Text Analysis Conference, TAC 2008, Gaithersburg, Maryland, USA, November 17-19, 2008*. URL: https://tac.nist.gov/publications/2008/additional.papers/RTE-4%5C_overview.proceedings.pdf.

Gillick, Dan, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh (2014). “Context-Dependent Fine-Grained Entity Type Tagging”. In: *CoRR* abs/1412.1820. arXiv: 1412.1820. URL: http://arxiv.org/abs/1412.1820.

Gionis, Aristides, Piotr Indyk, and Rajeev Motwani (1999). “Similarity Search in High Dimensions via Hashing”. In: *VLDB’99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pp. 518–529. URL: http://www.vldb.org/conf/1999/P49.pdf.

Glickman, Oren, Ido Dagan, and Moshe Koppel (2005). “A Probabilistic Classification Approach for Lexical Textual Entailment”. In: *Proceedings, The Twentieth National*

Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA, pp. 1050–1055. URL: `http://www.aaai.org/Library/AAAI/2005/aaai05-166.php`.

Goodman, Noah D and Daniel Lassiter (2015). "Probabilistic semantics and pragmatics: Uncertainty in language and thought". In: *The handbook of contemporary semantic theory, 2nd edition. Wiley-Blackwell.*

Gooi, Chung Heong and James Allan (2004). "Cross-Document Coreference on a Large Scale Corpus". In: *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004, Boston, Massachusetts, USA, May 2-7, 2004*, pp. 9–16. URL: `https://www.aclweb.org/anthology/N04-1002/`.

Graff, David (2002). "The AQUAINT Corpus of English News Text LDC2002T31". In: *Linguistic Data Consortium.*

Greenwood, Mark A., ed. (Aug. 2008). *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*. Manchester, UK. URL: `https://www.aclweb.org/anthology/W08-1800`.

Griffitt, Kira, Jennifer Tracey, Ann Bies, and Stephanie M. Strassel (2018). "Simple Semantic Annotation and Situation Frames: Two Approaches to Basic Text Understanding in LORELEI". In: *Proceedings of the Eleventh International Conference on Language*

Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018. URL: http://www.lrec-conf.org/proceedings/lrec2018/summaries/1020.html.

Gupta, Nitish, Sameer Singh, and Dan Roth (2017). “Entity Linking via Joint Encoding of Types, Descriptions, and Context”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 2681–2690. URL: https://doi.org/10.18653/v1/d17-1284.

Gururangan, Suchin, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith (2018). “Annotation Artifacts in Natural Language Inference Data”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pp. 107–112. URL: https://doi.org/10.18653/v1/n18-2017.

Guu, Kelvin, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang (2020). “REALM: Retrieval-augmented language model pre-training”. In: *Proceedings of the 37nd International Conference on Machine Learning, ICML 2020, Online, July 13-18, 2020*, pp. 5695–5704. URL: https://proceedings.icml.cc/static/paper_files/icml/2020/3102-Paper.pdf.

Haghighi, Aria and Dan Klein (2010). “Coreference Resolution in a Modular, Entity-Centered Model”. In: *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4,* *2010, Los Angeles, California, USA*, pp. 385–393. URL: https://www.aclweb.org/anthology/N10-1061/.

Han, Xu, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li (2018). “OpenKE: An Open Toolkit for Knowledge Embedding”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pp. 139–144. URL: https://www.aclweb.org/anthology/D18-2024/.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant (Sept. 2020). “Array programming with NumPy”. In: *Nature* 585.7825, pp. 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2.

Heilman, Michael and Noah A. Smith (2010). “Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions”. In: *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pp. 1011–1019. URL: https://www.aclweb.org/anthology/N10-1145/.

Herbrich, Ralf, Thore Graepel, and Klaus Obermayer (1999). “Support vector learning for ordinal regression”. In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. Vol. 1, pp. 97–102. URL: https://ieeexplore.ieee.org/document/819548.

Huang, Po-Sen, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck (2013). “Learning deep structured semantic models for web search using clickthrough data”. In: *22nd ACM International Conference on Information and Knowledge Management, CIKM’13, San Francisco, CA, USA, October 27 - November 1, 2013*, pp. 2333–2338. URL: https://doi.org/10.1145/2505515.2505665.

Imran, Muhammad, Carlos Castillo, Ji Lucas, Patrick Meier, and Sarah Vieweg (2014). “AIDR: artificial intelligence for disaster response”. In: *23rd International World Wide Web Conference, WWW ’14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume*, pp. 159–162. URL: https://doi.org/10.1145/2567948.2577034.

Iyer, Shankar, Nikhil Dandekar, and Kornél Csernai (2017). *First Quora Dataset Release: Question Pairs*. URL: https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

Jansen, Aren, Kenneth Church, and Hynek Hermansky (2010). “Towards spoken term discovery at scale with zero resources”. In: *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pp. 1676–1679. URL: http://www.isca-speech.org/archive/interspeech%5C_2010/i10%5C_1676.html.

Jansen, Aren and Benjamin Van Durme (2011). “Efficient spoken term discovery using randomized algorithms”. In: *2011 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU 2011, Waikoloa, HI, USA, December 11-15, 2011*, pp. 401–406. URL: https://doi.org/10.1109/ASRU.2011.6163965.

Jégou, Hervé, Matthijs Douze, and Cordelia Schmid (2011). “Product Quantization for Nearest Neighbor Search”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.1, pp. 117–128. URL: https://doi.org/10.1109/TPAMI.2010.57.

Jin, Hailong, Lei Hou, Juanzi Li, and Tiansi Dong (2019). “Fine-Grained Entity Typing via Hierarchical Multi Graph Convolutional 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 2019, Hong Kong, China, November 3-7, 2019*, pp. 4968–4977. URL: https://www.aclweb.org/anthology/D19-1502/.

Joachims, Thorsten (2002). “Optimizing search engines using clickthrough data”. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pp. 133–142. URL: https://doi.org/10.1145/775047.775067.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (2017). “Billion-scale similarity search with GPUs”. In: *CoRR* abs/1702.08734. arXiv: 1702.08734. URL: http://arxiv.org/abs/1702.08734.

Jones, Karen Spärck (1972). "A statistical interpretation of term specificity and its application in retrieval". In: *Journal of Documentation* 28.5, pp. 11–21. URL: https://doi.org/10.1108/eb026526.

Joshi, Mandar, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy (2020). "SpanBERT: Improving Pre-training by Representing and Predicting Spans". In: *Trans. Assoc. Comput. Linguistics* 8, pp. 64–77. URL: https://transacl.org/ojs/index.php/tacl/article/view/1853.

Joshi, Mandar, Omer Levy, Luke Zettlemoyer, and Daniel S. Weld (2019). "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, Hong Kong, China, November 3-7, 2019*, pp. 5802–5807. URL: https://doi.org/10.18653/v1/D19-1588.

Joulin, Armand, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave (2018). "Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2979–2984. URL: https://doi.org/10.18653/v1/d18-1330.

Karpukhin, Vladimir, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih (2020). "Dense Passage Retrieval for Open-Domain Question Answering”. In: *CoRR* abs/2004.04906. arXiv: 2004.04906. URL: https://arxiv.org/abs/2004.04906.

Lai, Alice and Julia Hockenmaier (2014). “Illinois-LH: A Denotational and Distributional Approach to Semantics”. In: *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*, pp. 329–334. URL: https://doi.org/10.3115/v1/s14-2055.

Lai, Alice and Julia Hockenmaier (2017). “Learning to Predict Denotational Probabilities For Modeling Entailment”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pp. 721–730. URL: https://doi.org/10.18653/v1/e17-1068.

Lai, Tuan Manh, Quan Hung Tran, Trung Bui, and Daisuke Kihara (2019). “A Gated Self-attention Memory Network for Answer Selection”. 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, Hong Kong, China, November 3-7, 2019*, pp. 5952–5958. URL: https://doi.org/10.18653/v1/D19-1610.

Lalor, John P., Hao Wu, Tsendsuren Munkhdalai, and Hong Yu (2018). “Understanding Deep Learning Performance through an Examination of Test Set Difficulty: A Psychometric Case Study”. In: *Proceedings of the 2018 Conference on Empirical Methods* *in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 4711–4716. URL: https://doi.org/10.18653/v1/d18-1500.

Lalor, John P., Hao Wu, and Hong Yu (2016). "Building an Evaluation Scale using Item Response Theory". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 648–657. URL: https://doi.org/10.18653/v1/d16-1062.

Lample, Guillaume, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou (2018). "Word translation without parallel data". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. URL: https://openreview.net/forum?id=H196sainb.

Lee, Kenton, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer (2015). "Event Detection and Factuality Assessment with Non-Expert Supervision". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 1643–1648. URL: https://doi.org/10.18653/v1/d15-1189.

Lee, Kenton, Ming-Wei Chang, and Kristina Toutanova (2019). "Latent Retrieval for Weakly Supervised Open Domain Question Answering". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 6086–6096. URL: https://doi.org/10.18653/v1/p19-1612.

Lee, Kenton, Luheng He, Mike Lewis, and Luke Zettlemoyer (2017). "End-to-end Neural Coreference Resolution". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 188–197. URL: https://doi.org/10.18653/v1/d17-1018.

Lee, Kenton, Luheng He, and Luke Zettlemoyer (2018). "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, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pp. 687–692. URL: https://doi.org/10.18653/v1/n18-2108.

Levesque, Hector J. (2011). "The Winograd Schema Challenge". In: *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. URL: http://www.aaai.org/ocs/index.php/SSS/SSS11/paper/view/2502.

Levesque, Hector J., Ernest Davis, and Leora Morgenstern (2012). "The Winograd Schema Challenge". In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. URL: http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4492.

Lew, Michael S., Nicu Sebe, Chabane Djeraba, and Ramesh C. Jain (2006). "Content-based multimedia information retrieval: State of the art and challenges". In: *ACM Trans.*

Multim. Comput. Commun. Appl. 2.1, pp. 1–19. URL: https://doi.org/10.1145/1126004.1126005.

Li, Hang (2014). *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers. URL: https://doi.org/10.2200/S00607ED2V01Y201410HLT026.

Li, Xiang, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum (2019). "Smoothing the Geometry of Probabilistic Box Embeddings". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=H1xSNiRcF7.

Li, Zhongyang, Tongfei Chen, and Benjamin Van Durme (2019). "Learning to Rank for Plausible Plausibility". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4818–4823. URL: https://doi.org/10.18653/v1/p19-1475.

Lin, Jimmy J. and Boris Katz (2006). "Building a reusable test collection for question answering". In: *J. Assoc. Inf. Sci. Technol.* 57.7, pp. 851–861. URL: https://doi.org/10.1002/asi.20348.

Lin, Ying and Heng Ji (2019). "An Attentive Fine-Grained Entity Typing Model with Latent Type Representation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natu-*

ral Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pp. 6196–6201. URL: https://doi.org/10.18653/v1/D19-1641.

Ling, Xiao and Daniel S. Weld (2012). “Fine-Grained Entity Recognition”. In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5152.

López, Federico, Benjamin Heinzerling, and Michael Strube (2019). “Fine-Grained Entity Typing in Hyperbolic Space”. In: *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*. Association for Computational Linguistics, pp. 169–180. URL: https://doi.org/10.18653/v1/w19-4319.

Loshchilov, Ilya and Frank Hutter (2019). “Decoupled Weight Decay Regularization”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

Luhn, Hans Peter (1957). “A Statistical Approach to Mechanized Encoding and Searching of Literary Information”. In: *IBM J. Res. Dev.* 1.4, pp. 309–317. URL: https://doi.org/10.1147/rd.14.0309.

Luo, Xiaoqiang (2005). “On Coreference Resolution Performance Metrics”. In: *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference,* *6-8 October 2005, Vancouver, British Columbia, Canada*, pp. 25–32. URL: https://www.aclweb.org/anthology/H05-1004/.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning (2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 1412–1421. URL: https://www.aclweb.org/anthology/D15-1166/.

Mann, Gideon S. and David Yarowsky (2003). "Unsupervised Personal Name Disambiguation". In: *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pp. 33–40. URL: https://www.aclweb.org/anthology/W03-0405/.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky (2014). "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pp. 55–60. URL: https://doi.org/10.3115/v1/p14-5010.

Marelli, Marco, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli (2014). *The SICK (Sentences Involving Compositional Knowledge) dataset for relatedness and entailment*. URL: https://doi.org/10.5281/zenodo.2787612.

Martschat, Sebastian and Michael Strube (2015). “Latent Structures for Coreference Resolution”. In: *Trans. Assoc. Comput. Linguistics* 3, pp. 405–418. URL: `https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/604`.

Mayfield, James, David Alexander, Bonnie J. Dorr, Jason Eisner, Tamer Elsayed, Tim Finin, Clayton Fink, Marjorie Freedman, Nikesh Garera, Paul McNamee, Saif Mohammad, Douglas W. Oard, Christine D. Piatko, Asad B. Sayeed, Zareen Syed, Ralph M. Weischedel, Tan Xu, and David Yarowsky (2009). “Cross-Document Coreference Resolution: A Key Technology for Learning by Reading”. In: *Learning by Reading and Learning to Read, Papers from the 2009 AAAI Spring Symposium, Technical Report SS-09-07, Stanford, California, USA, March 23-25, 2009*, pp. 65–70. URL: `http://www.aaai.org/Library/Symposia/Spring/2009/ss09-07-011.php`.

Metzler, Donald and W. Bruce Croft (2007). “Linear feature-based models for information retrieval”. In: *Inf. Retr.* 10.3, pp. 257–274. URL: `https://doi.org/10.1007/s10791-006-9019-z`.

Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin (2018). “Advances in Pre-Training Distributed Word Representations”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. URL: `http://www.lrec-conf.org/proceedings/lrec2018/summaries/721.html`.

Moldovan, Dan I., Christine Clark, Sanda M. Harabagiu, and Daniel Hodges (2007). “Cogex: A semantically and contextually enriched logic prover for question answering”.

In: *J. Appl. Log.* 5.1, pp. 49–69. URL: https://doi.org/10.1016/j.jal.2005.12.005.

Montague, Richard (1973). “The proper treatment of quantification in ordinary English”. In: *Approaches to natural language*. Springer, pp. 221–242.

Mostafazadeh, Nasrin, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen (2016). “A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories”. In: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pp. 839–849. URL: https://doi.org/10.18653/v1/n16-1098.

Mou, Lili, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin (2016). “Natural Language Inference by Tree-Based Convolution and Heuristic Matching”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. URL: https://doi.org/10.18653/v1/p16-2022.

Murty, Shikhar, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum (2018). “Hierarchical Losses and New Resources for Fine-grained Entity Typing and Linking”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 97–109. URL: https://www.aclweb.org/anthology/P18-1010/.

Nallapati, Ramesh (2004). “Discriminative models for information retrieval”. In: *SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 25-29, 2004*, pp. 64–71. URL: https://doi.org/10.1145/1008992.1009006.

Neyshabur, Behnam and Nathan Srebro (2015). “On Symmetric and Asymmetric LSHs for Inner Product Search”. In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1926–1934. URL: http://proceedings.mlr.press/v37/neyshabur15.html.

Ng, Vincent (2010). “Supervised Noun Phrase Coreference Research: The First Fifteen Years”. In: *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pp. 1396–1411. URL: https://www.aclweb.org/anthology/P10-1142/.

Ng, Vincent and Claire Cardie (2002). “Improving Machine Learning Approaches to Coreference Resolution”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pp. 104–111. URL: https://www.aclweb.org/anthology/P02-1014/.

Nistér, David and Henrik Stewénius (2006). “Scalable Recognition with a Vocabulary Tree”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pp. 2161–2168. URL: https://doi.org/10.1109/CVPR.2006.264.

Okuno, Akifumi, Geewook Kim, and Hidetoshi Shimodaira (2019). “Graph Embedding with Shifted Inner Product Similarity and Its Improved Approximation Capability”. In: *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pp. 644–653. URL: http://proceedings.mlr.press/v89/okuno19a.html.

Onoe, Yasumasa and Greg Durrett (2019). “Learning to Denoise Distantly-Labeled Data for Entity Typing”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 2407–2417. URL: https://www.aclweb.org/anthology/N19-1250/.

Park, A. S. and James R. Glass (2008). “Unsupervised Pattern Discovery in Speech”. In: *IEEE Trans. Speech Audio Process.* 16.1, pp. 186–197. URL: https://doi.org/10.1109/TASL.2007.909282.

Pavlick, Ellie and Chris Callison-Burch (2016). “Most “babies” are “little” and most “problems” are “huge”: Compositional Entailment in Adjective-Nouns”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. URL: https://doi.org/10.18653/v1/p16-1204.

Pavlick, Ellie and Tom Kwiatkowski (2019). “Inherent Disagreements in Human Textual Inferences”. In: *Trans. Assoc. Comput. Linguistics* 7, pp. 677–694. URL: `https://transacl.org/ojs/index.php/tacl/article/view/1780`.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018). “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 2227–2237. URL: `https://www.aclweb.org/anthology/N18-1202/`.

Phang, Jason, Thibault Févry, and Samuel R. Bowman (2018). “Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks”. In: *CoRR* abs/1811.01088. arXiv: `1811.01088`. URL: `http://arxiv.org/abs/1811.01088`.

Poliak, Adam, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme (2018). “Hypothesis Only Baselines in Natural Language Inference”. In: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 5-6, 2018*, pp. 180–191. URL: `https://doi.org/10.18653/v1/s18-2023`.

Ponte, Jay M. and W. Bruce Croft (1998). “A Language Modeling Approach to Information Retrieval”. In: *SIGIR ’98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-*

28 1998, Melbourne, Australia, pp. 275–281. URL: https://doi.org/10.1145/290941.291008.

Pradhan, Sameer, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang (2012). “CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes”. In: *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning - Proceedings of the Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes, EMNLP-CoNLL 2012, July 13, 2012, Jeju Island, Korea*, pp. 1–40. URL: https://www.aclweb.org/anthology/W12-4501/.

Raiman, Jonathan and Olivier Raiman (2018). “DeepType: Multilingual Entity Linking by Neural Type System Evolution”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 5406–5413. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17148.

Rao, Delip, Paul McNamee, and Mark Dredze (2010). “Streaming Cross Document Entity Coreference Resolution”. In: *COLING 2010, 23rd International Conference on Computational Linguistics, Posters Volume, 23-27 August 2010, Beijing, China*, pp. 1050–1058. URL: https://www.aclweb.org/anthology/C10-2121/.

Rao, Jinfeng, Hua He, and Jimmy J. Lin (2016). “Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks”. In: *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pp. 1913–1916. URL: https://doi.org/10.1145/2983323.2983872.

Ratinov, Lev-Arie and Dan Roth (2009). “Design Challenges and Misconceptions in Named Entity Recognition”. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL 2009, Boulder, Colorado, USA, June 4-5, 2009*. Ed. by Suzanne Stevenson and Xavier Carreras, pp. 147–155. URL: https://www.aclweb.org/anthology/W09-1119/.

Rehms, Georg and Hans Uszkoreit, eds. (2012). *META-NET White Paper Series: Europe’s languages in the digital age*. URL: www.meta-net.eu/whitepapers.

Ren, Xiang, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han (2016). “AFET: Automatic Fine-Grained Entity Typing by Hierarchical Partial-Label Embedding”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 1369–1378. URL: https://doi.org/10.18653/v1/d16-1144.

Ren, Xiang, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han (2016). “Label Noise Reduction in Entity Typing by Heterogeneous Partial-Label Embedding”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Dis-*

covery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pp. 1825–1834. URL: `https://doi.org/10.1145/2939672.2939822`.

Robertson, Stephen E. and Karen Spärck Jones (1976). "Relevance weighting of search terms". In: *J. Am. Soc. Inf. Sci.* 27.3, pp. 129–146. URL: `https://doi.org/10.1002/asi.4630270302`.

Robertson, Stephen E., Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford (1994). "Okapi at TREC-3". In: *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, pp. 109–126. URL: `http://trec.nist.gov/pubs/trec3/papers/city.ps.gz`.

Roemmele, Melissa, Cosmin Adrian Bejan, and Andrew S. Gordon (2011). "Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning". In: *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011.* URL: `http://www.aaai.org/ocs/index.php/SSS/SSS11/paper/view/2418`.

Roussopoulos, Nick, Stephen Kelley, and Frédéic Vincent (1995). "Nearest Neighbor Queries". In: *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA, May 22-25, 1995*, pp. 71–79. URL: `https://doi.org/10.1145/223784.223794`.

Rudinger, Rachel, Aaron Steven White, and Benjamin Van Durme (2018). "Neural Models of Factuality". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* *NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 731–744. URL: `https://doi.org/10.18653/v1/n18-1067`.

Sakaguchi, Keisuke, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi (2020). "WinoGrande: An Adversarial Winograd Schema Challenge at Scale". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. URL: `https://aaai.org/ojs/index.php/AAAI/article/view/6399`.

Sakaguchi, Keisuke and Benjamin Van Durme (2018). "Efficient Online Scalar Annotation with Bounded Support". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 208–218. URL: `https://www.aclweb.org/anthology/P18-1020/`.

Sankepally, Rashmi, Tongfei Chen, Benjamin Van Durme, and Douglas W. Oard (2018). "A Test Collection for Coreferent Mention Retrieval". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pp. 1209–1212. URL: `https://doi.org/10.1145/3209978.3210139`.

Schölkopf, Bernhard (2000). "The Kernel Trick for Distances". In: *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems*

(NIPS) 2000, Denver, CO, USA, pp. 301–307. URL: http://papers.nips.cc/paper/1862-the-kernel-trick-for-distances.

Severyn, Aliaksei and Alessandro Moschitti (2015). “Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks”. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pp. 373–382. URL: https://doi.org/10.1145/2766462.2767738.

Shapiro, Amram, Louise Firth Campbell, and Rosalind Wright (2014). *Book of Odds. From Lightning Strikes to Love at First Sight, the Odds of Everyday Life*. William Morrow Paperbacks.

Shimaoka, Sonse, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel (2016). “An Attentive Neural Architecture for Fine-grained Entity Type Classification”. In: *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pp. 69–74. URL: http://aclweb.org/anthology/W/W16/W16-1313.pdf.

Shimaoka, Sonse, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel (2017). “Neural Architectures for Fine-grained Entity Type Classification”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pp. 1271–1280. URL: https://doi.org/10.18653/v1/e17-1119.

Shrivastava, Anshumali and Ping Li (2014). “Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS)”. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2321–2329. URL: http://papers.nips.cc/paper/5329-asymmetric-lsh-alsh-for-sublinear-time-maximum-inner-product-search-mips.

Silpa-Anan, Chanop and Richard I. Hartley (2008). “Optimised KD-trees for fast image descriptor matching”. In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. URL: https://doi.org/10.1109/CVPR.2008.4587638.

Smeulders, Arnold W. M., Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh C. Jain (2000). “Content-Based Image Retrieval at the End of the Early Years”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.12, pp. 1349–1380. URL: https://doi.org/10.1109/34.895972.

Stanovsky, Gabriel, Judith Eckle-Kohler, Yevgeniy Puzikov, Ido Dagan, and Iryna Gurevych (2017). “Integrating Deep Linguistic Features in Factuality Prediction over Unified Datasets”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pp. 352–357. URL: https://doi.org/10.18653/v1/P17-2056.

Stoyanov, Veselin and Jason Eisner (2012). “Easy-first Coreference Resolution”. In: *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings* *of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pp. 2519–2534. URL: https://www.aclweb.org/anthology/C12-1154/.

Strassel, Stephanie M., Ann Bies, and Jennifer Tracey (2017). "Situational Awareness for Low Resource Languages: the LORELEI Situation Frame Annotation Task". In: *Proceedings of the First International Workshop on Exploitation of Social Media for Emergency Relief and Preparedness co-located with European Conference on Information Retrieval, SMERP@ECIR 2017, Aberdeen, UK, April 9, 2017*, pp. 32–41. URL: http://ceur-ws.org/Vol-1832/SMERP%5C_2017%5C_peer%5C_review%5C_paper%5C_5.pdf.

Strassel, Stephanie M. and Jennifer Tracey (2016). "LORELEI Language Packs: Data, Tools, and Resources for Technology Development in Low Resource Languages". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. URL: http://www.lrec-conf.org/proceedings/lrec2016/summaries/1138.html.

Tan, Ming, Bing Xiang, and Bowen Zhou (2015). "LSTM-based Deep Learning Models for non-factoid answer selection". In: *CoRR* abs/1511.04108. arXiv: 1511.04108. URL: http://arxiv.org/abs/1511.04108.

Teichert, Adam R., Adam Poliak, Benjamin Van Durme, and Matthew R. Gormley (2017). "Semantic Proto-Role Labeling". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 4459–4466. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14997.

Trouillon, Théo, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard (2016). “Complex Embeddings for Simple Link Prediction”. In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2071–2080. URL: http://proceedings.mlr.press/v48/trouillon16.html.

Tsochantaridis, Ioannis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun (2005). “Large Margin Methods for Structured and Interdependent Output Variables”. In: *J. Mach. Learn. Res.* 6, pp. 1453–1484. URL: http://jmlr.org/papers/v6/tsochantaridis05a.html.

Tsuchiya, Masatoshi (2018). “Performance Impact Caused by Hidden Bias of Training Data for Recognizing Textual Entailment”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. URL: http://www.lrec-conf.org/proceedings/lrec2018/summaries/786.html.

Tversky, Amos and Daniel Kahneman (1981). “The framing of decisions and the psychology of choice”. In: *Science* 211.4481, pp. 453–458.

van Deemter, Kees and Rodger Kibble (2000). “On Coreferring: Coreference in MUC and Related Annotation Schemes”. In: *Comput. Linguistics* 26.4, pp. 629–637. URL: https://www.aclweb.org/anthology/J00-4005/.

Van Eijck, Jan and Shalom Lappin (2012). “Probabilistic semantics for natural language”. In: *Logic and interactive rationality (LIRA)* 2, pp. 17–35.

Vilain, Marc B., John D. Burger, John S. Aberdeen, Dennis Connolly, and Lynette Hirschman (1995). “A model-theoretic coreference scoring scheme”. In: *Proceedings of the 6th Conference on Message Understanding, MUC 1995, Columbia, Maryland, USA, November 6-8, 1995*, pp. 45–52. URL: https://doi.org/10.3115/1072399.1072405.

Vilnis, Luke, Xiang Li, Shikhar Murty, and Andrew McCallum (2018). “Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Ed. by Iryna Gurevych and Yusuke Miyao. Association for Computational Linguistics, pp. 263–272. URL: https://www.aclweb.org/anthology/P18-1025/.

Vulić, Ivan, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen (2017). “HyperLex: A Large-Scale Evaluation of Graded Lexical Entailment”. In: *Computational Linguistics* 43.4. URL: https://doi.org/10.1162/COLI%5C_a%5C_00301.

Wan, Ji, Dayong Wang, Steven Chu-Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li (2014). “Deep Learning for Content-Based Image Retrieval: A Comprehensive Study”. In: *Proceedings of the ACM International Conference on Multimedia, MM ’14, Orlando, FL, USA, November 03 - 07, 2014*, pp. 157–166. URL: https://doi.org/10.1145/2647868.2654948.

Wang, Di and Eric Nyberg (2015). “A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pp. 707–712. URL: https://doi.org/10.3115/v1/p15-2116.

Wang, Mengqiu and Christopher D. Manning (2010). “Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering”. In: *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, pp. 1164–1172. URL: https://www.aclweb.org/anthology/C10-1131/.

Wang, Mengqiu, Noah A. Smith, and Teruko Mitamura (2007). “What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA”. In: *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pp. 22–32. URL: https://www.aclweb.org/anthology/D07-1003/.

Weischedel, Ralph and Ada Brunstein (2005). “BBN Pronoun Coreference and Entity Type Corpus”. In: *Philadelphia: Linguistic Data Consortium*. URL: https://catalog.ldc.upenn.edu/LDC2005T33.

Weischedel, Ralph, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston (2013). “OntoNotes Release 5.0”. In: *Philadelphia: Linguistic Data Consortium*. URL: https://catalog.ldc.upenn.edu/LDC2013T19.

Weston, Jason and Chris Watkins (1999). “Support vector machines for multi-class pattern recognition”. In: *ESANN 1999, 7th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 21-23, 1999, Proceedings*, pp. 219–224. URL: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es1999-461.pdf.

Williams, Adina, Nikita Nangia, and Samuel R. Bowman (2018). “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 1112–1122. URL: https://doi.org/10.18653/v1/n18-1101.

Wiseman, Sam, Alexander M. Rush, and Stuart M. Shieber (2016). “Learning Global Features for Coreference Resolution”. In: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pp. 994–1004. URL: https://doi.org/10.18653/v1/n16-1114.

Wiseman, Sam, Alexander M. Rush, Stuart M. Shieber, and Jason Weston (2015). “Learning Anaphoricity and Antecedent Ranking Features for Coreference Resolution”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguis-*

tics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pp. 1416–1426. URL: `https://doi.org/10.3115/v1/p15-1137`.

Xiong, Wenhan, Jiawei Wu, Deren Lei, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang (2019). “Imposing Label-Relational Inductive Bias for Extremely Fine-Grained Entity Typing”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 773–784. URL: `https://www.aclweb.org/anthology/N19-1084/`.

Xu, Peng and Denilson Barbosa (2018). “Neural Fine-Grained Entity Type Classification with Hierarchy-Aware Loss”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 16–25. URL: `https://www.aclweb.org/anthology/N18-1002`.

Yaghoobzadeh, Yadollah and Hinrich Schütze (2015). “Corpus-level Fine-grained Entity Typing Using Contextual Information”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal,* *September 17-21, 2015*, pp. 715–725. URL: https://doi.org/10.18653/v1/d15-1083.

Yang, Yi, Wen-tau Yih, and Christopher Meek (2015). “WikiQA: A Challenge Dataset for Open-Domain Question Answering”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 2013–2018. URL: https://doi.org/10.18653/v1/d15-1237.

Yao, Xuchen and Benjamin Van Durme (2014). “Information Extraction over Structured Data: Question Answering with Freebase”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pp. 956–966. URL: https://doi.org/10.3115/v1/p14-1090.

Yao, Xuchen, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark (2013). “Answer Extraction as Sequence Tagging with Tree Edit Distance”. In: *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pp. 858–867. URL: https://www.aclweb.org/anthology/N13-1106/.

Yao, Xuchen, Benjamin Van Durme, and Peter Clark (2013). “Automatic Coupling of Answer Extraction and Information Retrieval”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia,* *Bulgaria, Volume 2: Short Papers*, pp. 159–165. URL: https://www.aclweb.org/anthology/P13-2029/.

Yih, Wen-tau, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak (2013). “Question Answering Using Enhanced Lexical Semantic Models”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pp. 1744–1753. URL: https://www.aclweb.org/anthology/P13-1171/.

Yin, Wenpeng, Hinrich Schütze, Bing Xiang, and Bowen Zhou (2016). “ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs”. In: *Trans. Assoc. Comput. Linguistics* 4, pp. 259–272. URL: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/831.

Yogatama, Dani, Daniel Gillick, and Nevena Lazic (2015). “Embedding Methods for Fine Grained Entity Type Classification”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pp. 291–296. URL: https://www.aclweb.org/anthology/P15-2048/.

Yosef, Mohamed Amir, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum (2012). “HYENA: Hierarchical Type Classification for Entity Names”. In: *COLING 2012, 24th International Conference on Computational Linguistics, Proceed-*

ings of the Conference: Posters, 8-15 December 2012, Mumbai, India, pp. 1361–1370. URL: https://www.aclweb.org/anthology/C12-2133/.

Yu, Lei, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman (2014). “Deep Learning for Answer Sentence Selection”. In: *CoRR* abs/1412.1632. arXiv: 1412.1632. URL: http://arxiv.org/abs/1412.1632.

Zhai, ChengXiang and John D. Lafferty (2001). “A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval”. In: *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pp. 334–342. URL: https://doi.org/10.1145/383952.384019.

Zhang, Mozhi, Jordan Boyd-Graber, Michelle Yuan, C. Anton Rytting, Weiwei Yang, Philip Resnik, Ting Hua, Adam Poliak, Adam Teichert, Tongfei Chen, Xu Han, Linghao Jin, João Sedoc, and Benjamin Van Durme (2019). “LoReHLT19 System description UMD-JHU”. In: *LoReHLT19 System Descriptions*. URL: https://www.nist.gov/itl/iad/mig/lorehlt-evaluations.

Zhang, Sheng, Kevin Duh, and Benjamin Van Durme (2018). “Fine-grained Entity Typing through Increased Discourse Context and Adaptive Classification Thresholds”. In: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 5-6, 2018*, pp. 173–179. URL: https://doi.org/10.18653/v1/s18-2022.

Zhang, Sheng, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme (2017). “Ordinal Common-sense Inference”. In: *Trans. Assoc. Comput. Linguistics* 5, pp. 379–395. URL: `https://transacl.org/ojs/index.php/tacl/article/view/1082`.

Vita

Tongfei Chen graduated with a B.Sc. in Computer Science from Peking University, China in 2014. His research focuses on computational semantics and dialogue systems, and developing ranking approaches to these problems to address their uncertainty, and retrieval algorithms for these that leads to scalability. He has interned as research scientists at IBM T. J. Watson Research Center, and Alexa AI at Amazon.com, Inc., where a work from the latter internship won a Best Paper Award at the First Workshop in NLP for Conversational AI, at Florence, Italy.