INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAT017

Thèse de doctorat

TELECOM Paris

IP PARIS

Towards Efficient, General and Robust Entity Disambiguation Systems

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Palaiseau, le 14 Juin 2023, par

LIHU CHEN

Composition du Jury :

Chloé Clavel Professeur, Institut Polytechnique de Paris	Présidente / Examinatrice
Evangelos Kanoulas Professeur, Université d'Amsterdam	Rapporteur
Gerard de Melo Professeur, Hasso Plattner Institute / Université de Potsdam	Rapporteur
Serena Villata Directrice de recherche, Université Côte d'Azur / CNRS	Examinatrice
Mrinmaya Sachan Assistant Professor, École polytechnique fédérale de Zurich	Examinateur
Fabian Suchanek Professeur, Institut Polytechnique de Paris	Directeur de thèse
Gaël Varoquaux Directeur de Recherche, Inria Saclay	Co-Directeur de thèse

PHD THESIS

Towards Efficient, General and Robust Entity Disambiguation Systems

Author:
Lihu CHEN

TÉLÉCOM PARIS
Institut Polytechnique de Paris

August 26, 2023

Abrégé

La désambiguïsation des entités vise à faire correspondre les mentions dans les documents à des entités standard dans une base de connaissances donnée, ce qui est important pour diverses applications telles que l'extraction d'informations, la recherche sur le web et la réponse aux questions. Bien que le domaine soit très dynamique et que de nombreux travaux nouveaux apparaissent, trois questions sont sous-explorées par les travaux antérieurs.

1) Peut-on utiliser un petit modèle pour approcher les performances d'un grand modèle? Pour répondre à cette question, nous proposons un réseau neuronal léger mais réseau neuronal léger mais efficace pour la désambiguïsation d'entités biomédicales. Notre modèle est 23 Notre modèle est 23 fois plus petit et 6,4 fois plus rapide que le modèle de base BERT. empiriques démontrent que le modèle est très compétitif et atteint une performance qui est statistiquement indiscernable de l'état de l'art.

2) Comment développer un système de désambiguïsation unique adapté à plusieurs domaines? Pour répondre à cette question, nous présentons notre construction GLADIS, un nouveau benchmark pour la désambiguïsation d'acronymes, qui est plus grand et plus difficile que que les travaux existants. Ce benchmark contient trois éléments : un dictionnaire plus grand, trois ensembles de données provenant des domaines général, scientifique et biomédical, ainsi qu'un corpus de corpus de pré-entraînement à grande échelle. Nous avons également proposé AcroBERT, le premier modèle de langage pré-entraîné pour la désambiguïsation des acronymes. pour la désambiguïsation des acronymes, dont les performances sont nettement supérieures à celles d'autres modèles de référence dans de nombreux domaines

3) Les systèmes existants sont-ils robustes face aux mots hors-vocabulaire et aux différents ordres de mots? Pour la première sous-question, nous abordons la question de l'absence de vocabulaire en en concevant un cadre d'apprentissage contrastif, baptisé LOVE (Learning Out-of- Vocabulary Embeddings). LOVE est capable de générer des représentations de mots pour n'importe quel mot non vu en apprenant le comportement des mots non vus. mots non vus en apprenant le comportement d'encastrements pré-entraînés en utilisant uniquement la forme superficielle des mots. la forme su-

perficielle des mots. LOVE peut rendre les modèles de langage plus robustes avec quelques paramètres supplémentaires. Des évaluations approfondies démontrent que notre modèle léger performances similaires, voire supérieures, à celles de ses concurrents, à la fois sur des ensembles de données sur des ensembles de données originaux et sur des variantes corrompues. Pour la deuxième sous-question, nous soulignons une faiblesse potentielle faiblesse potentielle des encodages positionnels, qui sont largement utilisés dans les modèles basés sur les transformateurs. Les codages positionnels sont utilisés pour injecter des caractéristiques d'ordre des mots dans les modèles linguistiques. Bien qu'ils puissent Bien qu'ils puissent améliorer de manière significative les représentations de phrases, leur fonction spécifique dans les modèles de langage n'est pas entièrement comprise. aux modèles de langage n'est pas entièrement comprise, surtout si l'on tient compte des récentes la compréhension du langage naturel à partir de modèles de langage avec des encodages positionnels sont insensibles à l'ordre des mots. sont insensibles à l'ordre des mots. Dans ce travail, nous menons des études plus approfondies et systématiques sur les encodages positionnels. systématique des encodages positionnels, complétant ainsi les travaux existants sur deux aspects : Nous Nous révélons d'abord la fonction principale des PE en identifiant deux propriétés communes, la localité et la symétrie. la symétrie. Ensuite, nous mettons en évidence une faiblesse potentielle des PE actuels en introduisant deux nouvelles tâches d'évaluation de la permutation de mots. en introduisant deux nouvelles tâches d'évaluation de l'échange de mots. Nous espérons que ces nouveaux résultats et conclusions pourront nous éclairer sur la manière de concevoir et d'injecter des encodages positionnels dans les modèles de langage.

Enfin, nous présentons une nouvelle méthode d'achèvement de la KB basée sur LM, spécifiquement adaptée aux faits concernant les entités à longue queue. KB basée sur LM qui est spécifiquement conçue pour les faits concernant les entités à longue traîne. Plus précisément, nous présentons une nouvelle méthode de complétion de KB basée sur LM, spécifiquement des faits concernant des entités à longue traîne. La méthode exploite deux LM différents en deux étapes : pour la recherche de candidats et pour la vérification et la désambiguïsation des candidats. Pour évaluer notre méthode et diverses lignes de base, nous introduisons un nouvel ensemble de données, appelé MALT, enraciné dans Wikidata. dans Wikidata. Notre méthode surpasse toutes les lignes de base en F1, avec des gains importants, en particulier dans le rappel.

Acknowledgements

I spent three wonderful years in France and finally finished my PhD. I would like to thank many people who help me, support me and accompany me.

First of all, I want to thank my two thesis advisors. I would like to first thank my advisor at Telecom, Fabian. He is a knowledgeable, amiable, and altruistic person. Every time I meet him, he always talks with me with warmth and passion, and the topics of the talk can be very diverse, from traveling to culture, and we certainly discuss our research work as well. Fabian is very rigorous and responsible with our research work. At the same time, he highly emphasizes integrity and openness. I would say that I benefit a lot from working with Fabian. Next, I would like to thank my advisor at Inria, Gaël. He has a great passion and taste for research. He is able to give me suggestions from a very high-level view, which is the right thing that I am lacking. Gaël is committed to open-source projects and high-impact work, and this spirit inspires me a lot. It is great luck to be able to work with the two advisors, and I have obtained so much happiness, confidence, and achievements by working with them.

Secondly, I would like to thank my family members, my father, my mother, and especially my wife, Lixue. She is a sweet, kind, tender girl, and makes a warm, cozy and clean home for me. She always supports me and loves me. Without her, I would not be able to focus on my career and dreams. I appreciate her efforts very much.

Finally, I would like to thank my colleagues and friends. I have had great three years with all of you Diggers (people at the DIG team) and I enjoy all the lunchtime, seminars, and outings with you. Also, I would like to thank my friends for their accompany. I cherish the great memories of going to restaurants and playing football together, and you bring me a lot of joy.

Contents

List of Figures

List of Tables

1 Introduction

Entity-centric knowledge bases are large collections of facts about entities of public interest, such as countries, politicians, or movies [255]. A prominent usage of this type of knowledge is Web search, where we strive to retrieve factual information of entities, e.g., the birth date, spouse and education background of *"Joe Biden"*. For such queries, the key assets are *Knowledge Bases* (KBs), which contain large amounts of entity-centric facts such as the birth date, spouse, and graduate school of Joe Biden *<Joe Biden, date of birth, 1942-11-20>*, *<Joe Biden, spouse, Jill Biden>* and *<Joe Biden, graduated at, Syracuse University>*. Recently, an increasing number of large-scale and high-quality KBs have been constructed, such as Wikidata [274], DBpedia [9], YAGO [254], Freebase [21], and BabelNet [189]. These KBs contain millions of entities and their respective relational facts, which provide machine-readable knowledge. Bridging plain text and KBs is beneficial for a wide range of language understanding tasks like information extraction and question answering [238, 240]. However, natural language is notorious for its ambiguity: words or phrases (including sentences) have more than one meaning [78]. A concrete ambiguous example is below, which is a real sample from the question answering dataset TriviaQA [131]. This

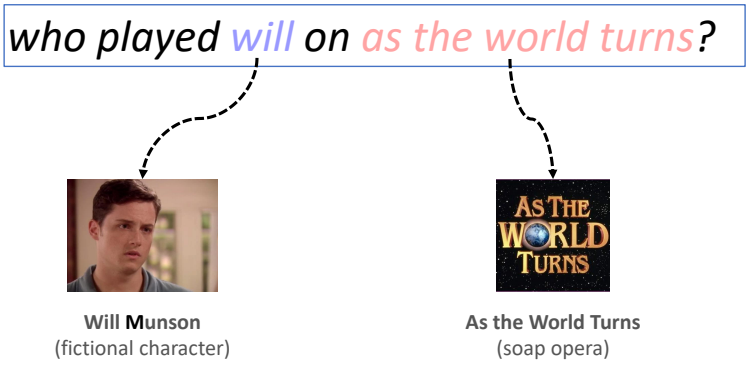

Figure 1.1: A question sample from TriviaQA [131]. The mention *"will"* and *"as the world turns"* means the fictional character Will Munson and the soap opera As the World Turns, respectively.

question wonders about the name of a specific actor, but machines may have difficulty understanding it due two the two ambiguous mentions *"will"* (blue) and *"as the world turns"* (red). "will" can be common names for male, song, film, etc. *"as the world turns"* can be a television soap opera, an album or a song. Actually, the first mention means a fictional character whose name is *"Will Munson"* and the second mention means an American television soap opera. Equipped with the entity information, we can better understand the semantics of the question.

In order to mitigate the gap between natural language and entities in KBs, one can use *Entity Linking* (EL), which aims to map mentions (or recognized entities) in documents to the canonicalized entities in a given KB. In general, there are two steps in EL: Named Entity Recognition (NER) and Entity Disambiguation (ED). The first step detects entity mentions in text documents, i.e., it detects mention *"Paris"*, while the next step maps them to standard entities in the KB, i.e., link the mention to the entity *"Paris (mythology)"*. The task of NER is already well-studied as shown in several surveys [187, 297, 160]. In addition, there are many publicly available NER tools and systems, such as spaCy[1], AllenNLP[2] and HuggingFace[3]. Since NER has already been explored well, in this work, we focus on entity disambiguation.

1.1 Entity Disambiguation

An example of entity disambiguation is shown in Figure 1.2. In the input text, there are four mentions: *Hurricane, Carter, Bob* and *Washington*, and our goal here is to link them to the correct entity in a given knowledge base. Note that each mention is ambiguous with multiple candidates. To construct the correct mapping, entity disambiguation systems commonly consider several signals that are related to the mentions and entity candidates, and we use the example to illustrate these signals briefly. The mention-entity popularity can offer some priori information for disambiguation. In this case, the mention *"Bob"* mostly denotes *Bob Dylan* because the American singer is famous around the world and the name *"Bob"* is highly associated with the entity. Besides, the Mention-Entity context similarity is also beneficial. For example, the context word *"tracks"* around the mention *"Hurricane"* reveals that it is probably related to a song. Moreover, the entity-entity coherence provides some useful clues. For example, the song Hurricane is a protest song by *Bob Dylan*, and its lyrics are about the Afro-American boxer *Rubin Carter*, who was wrongfully convicted for murder in the 1970s and later released after 20 years in prison. A system can obtain better performance by mapping mentions to a set of thematically self-consistent candidate entities. You can find the details of these features in Section 2.2. The entity disambiguation task has a long history of research with several surveys for references [241, 238, 240]. and it is important for many applications. Next, we briefly present several scenarios.

[1] https://spacy.io/api/entityrecognizer
[2] https://demo.allennlp.org/named-entity-recognition
[3] https://huggingface.co/models?sort=downloads&search=ner

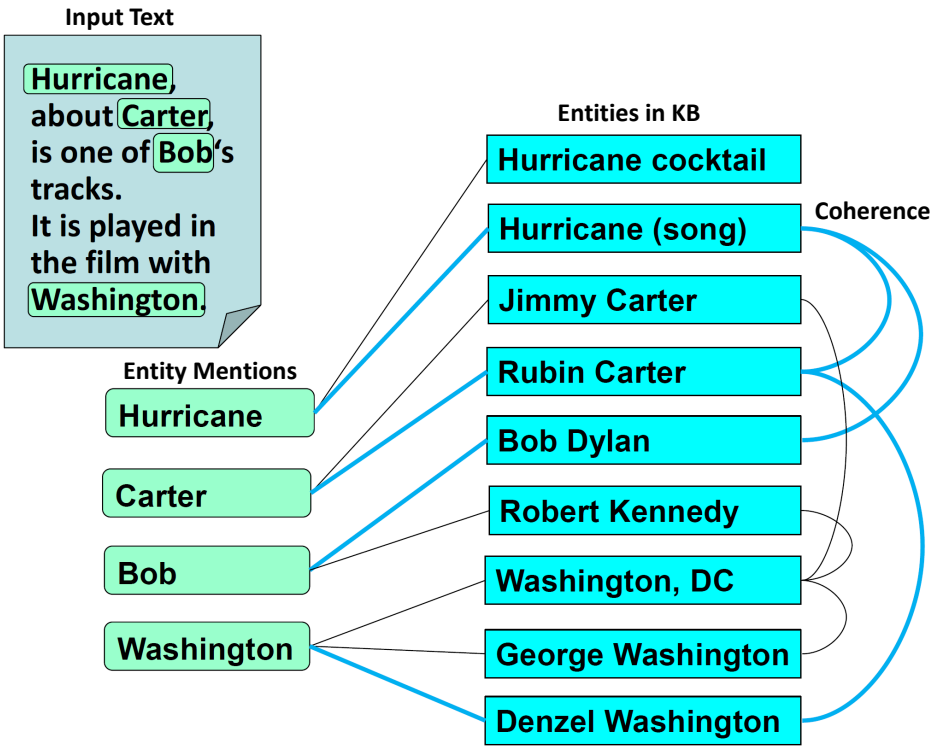

Figure 1.2: An example of entity disambiguation [284]. The green cells are four mentions found in the input text, and the blue cells are respective entity candidates. The lines between mentions and entities are correct labels. The lines among entities are the coherence strength, i.e., a semantic relationship between them.

Information Extraction (IE) is the task of automatically extracting structured information from unstructured or semi-structured documents. However, extracted entities or relations are usually ambiguous. Linking them to existing KBs needs entity disambiguation, which is essential for knowledge base completion [284] and population [120]. Given the sentence *"Paris is the son of King Priam"*, we can extract two entities *"Paris"* and *"King Priam"*, meanwhile, there is an extracted relation *"son of"* between the two entities. Suppose that we like to add this piece of fact into a newly constructed KB, but the KB already contains one Paris entity (the capital of France). Therefore, we need to create a new entity *Paris (mythology)* with a short description *"Paris is a mythological nobleman that appears in a number of Greek legends."*, and link the mention to it. Likewise, we repeat the operation for the other mention and also the relation. After, we insert a new fact to the KB successfully. Entity disambiguation is a key step for associating the extracted entities or relations to a KB, which can help enrich the scope and scale of KBs.

Question Answering (QA) aims to retrieve the answer to a question from a given text, which is useful for searching for answers in documents. Many question answering systems use a knowledge base to give answers to user questions. As shown in the example in Figure 1.1, question answering systems can benefit from entity disambiguation by better understanding questions.

Information Retrieval (IR) is the search for documents that meet information needs in unstructured text. Named entities usually appear in search queries and they are usually ambiguous [94]. For example, the entity mention *"Barcelona"* in the search query can mean many different entities, such as the capital city of Catalonia, the football club from Barcelona, a community in the United States and many songs and films whose names are *"Barcelona"*. Disambiguating mentions in queries is beneficial for search systems to locate relevant documents more precisely.

Knowledge-Enhanced Pre-trained Language Models. The reasoning power of pre-trained language models is also limited because they are purely data-driven. This drawback can be improved by incorporating external knowledge, which leads to knowledge-enhanced pre-trained language models. To inject knowledge into pre-trained language models, we might need entity disambiguation. For example, KnowBERT [206] uses an entity linker to retrieve relevant entity embeddings from a KB for enriching the entity-span representations. ERNIE [309] integrates entity representations, which are obtained by aligning mentions to their corresponding entities in KBs, into the underlying layers of transformer architecture for encoding knowledge information. Entity disambiguation is an effective measure to integrate knowledge into language models.

1.2 Challenges for Entity Disambiguation

Entity disambiguation is challenging due to the variation and the ambiguity of entity names. First, an entity in a KB may have many different surface forms, e.g., full name, nickname, partial name and abbreviation. For example, the entity Elvis Presley (`Q303`) has many names: *"Elvis Aaron Presley"*, *"King of Rock'n'Roll"*, *"Elvis"*, *"Elvis A. Presley"*, etc. Second, a mention in a document may refer to many different entities. For example, there are in total more than one hundred *"Paris"* in Wikipedia that cover places, people, media and so on. Moreover, considering that the scale of knowledge bases is continuously growing, e.g., the current Wikidata contains over 102M entities[4], it is increasingly difficult to perform entity disambiguation.

The field of entity disambiguation is vibrant with many novel works popping up, and recent works have developed new approaches for addressing the issues above [240, 238]. However, there are still three challenges that are underexplored by prior work:

Efficiency. The large pre-trained language models like GPT-X [219, 26] nowadays take a lead in various NLP tasks, and their sizes are growing larger and larger with

[4] https://www.wikidata.org/wiki/Special:Statistics

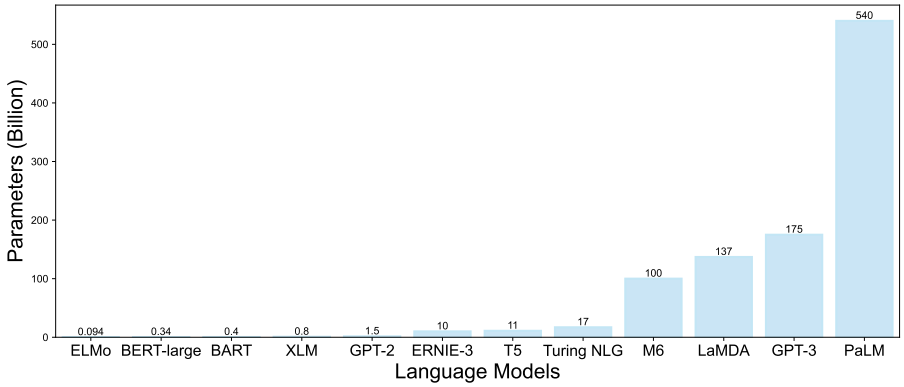

Figure 1.3: Comparison of the size of existing popular pre-trained language models. The size of models ranges from 0 to 540 billion.

millions or billions of parameters, as shown in Figure 1.3. The current state-of-the-art entity disambiguation models mostly use large pre-trained language models as the backbone [291, 57, 11]. To pre-train such language models, we need large-scale data and powerful computing resources, and the inference speed might be slow due to the large size, which is impractical for resource-limited users. Hence we ask that **Can we use a small model to approach the performance of a big model?**

Generalizability refers to the model's capability to adapt and react properly to previously unseen data. Most existing entity disambiguation systems target one particular knowledge base or one single specific domain, therefore, the performances of these systems deteriorate drastically when we apply them to another domain. For example, many systems use Wikipedia as the target KB [291, 57], but Wikipedia only contains a small fraction of biomedical concepts. Therefore, researchers prefer to design separate disambiguation systems for medical knowledge bases such as UMLS [19]. Likewise, the existing benchmarks for evaluation focus mainly on one domain, which make it difficult to develop a general system. A question naturally appears here: **How to develop a single disambiguation system adapted to multiple domains?**

Robustness. Although today's language models are exceptionally powerful, they are not robust enough. Liang et al. [162] show that text classifiers based on deep learning can be easily fooled by small character-level perturbations. Sun et al. [257] demonstrate that the BERT model is not robust when dealing with misspellings. Boukkouri et al. [68] point out that the BERT does not output good representations for certain medical words. These phenomena indicate that some language models are not robust enough for the **Out-of-Vocabulary** (OOV) problem. For example, the first sentence below is a movie comment from a real-world dataset. If we fine-tune BERT on it, then BERT can know this sample is a positive comment. However, if we slightly modify the word *"successful"* by removing one s, BERT then classifies it as negative, which means small perturbations can flip the prediction of advanced models. Terribly, some malicious

people may exploit this weakness to attack NLP systems such as the systems of spam check or hate speech detection.

> *1) altogether, this is successful as a film*
> *2) altogether, this is succesful as a film*

On the other hand, many pre-trained language models are not sensitive to word orders, which are essential for understanding languages. For example, the sentence pair below from SNLI [23] satisfies the entailment relation (the semantic of the second sentence can be derived from the inference of the first sentence):

> *1) A man playing an electric guitar on stage*
> *2) A man playing guitar on stage*

If we change the word order of the premise sentence so that it becomes *"an electric guitar playing a man"*. Then the relationship between the two sentences is neutral or even contradicted, but we surprisingly found that the predicted result of a fine-tuned LM does not change, which shows that some existing language models are not robust to the swap of **Word Orders**. Because current entity disambiguation systems highly rely on language models, we may ask: **Can we make these systems more robust?**

1.2.1 Contributions

In Chapter 3, we answer the question about efficiency by looking at the task of disambiguating biomedical entities. Concretely, we present a lightweight yet effective neural model for biomedical entity disambiguation. Our experimental results on three biomedical evaluation benchmarks show that the model is very effective, and achieves a performance that is statistically indistinguishable from the state-of-the-art. BERT-based models, e.g., have 23 times more parameters and require 6.4 times more computing time for inference. Chapter 3 is based on the paper [38]:

> *Lihu Chen, Gaël Varoquaux, and Fabian M. Suchanek. "**A Lightweight Neural Model for Biomedical Entity Linking**." (Full Paper) Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 14. 2021.*

In Chapter 4, We reply to the question about generalizability by focusing on the task of disambiguation of acronyms. Specifically, we construct GLADIS, a challenging benchmark for Acronym Disambiguation, which includes a larger dictionary, three datasets from the general, scientific, and biomedical domains, and a large-scale pre-training corpus. We also propose AcroBERT, a BERT-based model that is pre-trained on our collected acronym documents, which can significantly outperform other baselines across multiple domains, and which is more robust in the presence of very ambiguous acronyms and overshadowed samples. Chapter 4 is based on the paper [40]:

> *Lihu Chen, Gaël Varoquaux, and Fabian M. Suchanek. "**GLADIS: A General and Large Acronym Disambiguation Benchmark**." (Full Paper) Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. 2023*

In Chapter 5, we answer the question about robustness by addressing the out-of-vocabulary (OOV) problem. In this part, we follow the principle of mimick-like models to generate vectors for unseen words, by learning the behavior of pre-trained embeddings using only the surface form of words. We present a simple contrastive learning framework, LOVE, which extends the word representation of an existing pre-trained language model (such as BERT), and makes it robust to the OOV with few additional parameters. Extensive evaluations demonstrate that our lightweight model achieves similar or even better performances than prior competitors, both on original datasets and on corrupted variants. Moreover, it can be used in a plug-and-play fashion with FastText and BERT, where it significantly improves their robustness. Chapter 5 is based on the paper [37]:

> *Lihu Chen, Gaël Varoquaux, and Fabian M. Suchanek. "**Imputing Out-of-Vocabulary Embeddings with LOVE Makes LanguageModels Robust with Little Cost**." (Full Paper) Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022.*

In Chapter 6, we continue our discussion of robustness by identifying a potential flaw in positional encodings. Positional Encodings (PEs) are used to infuse word-order information into transformer-based language models. Although they can significantly enhance sentence representations, their specific function for language models are not fully understood, especially given recent findings that building natural-language understanding from language models with positional encodings are insensitive to word order. In this work, we conduct more in-depth and systematic studies of positional encodings, thus complementing existing work in two aspects: We first reveal the core function of PEs by identifying two common properties, Locality and Symmetry. Then, we first point out a potential weakness of current PEs by introducing two new probing tasks of word swap. We hope these new probing results and findings can shed light on how to design and inject positional encodings into language models. Chapter 6 is based on the paper [39]:

> *Lihu Chen, Gaël Varoquaux, and Fabian M. Suchanek. "**Understanding The Role of Positional Encodings in Sentence Representations**." (Full Paper) In Progress.*

In Chapter 7, we explore how to use current entity disambiguation model for knowledge base completion.Despite their impressive size, knowledge bases (KBs) are still far from complete. Language models (LMs) have been proposed as a source to fill these gaps. However, prior work has focused on salient entities with rich coverage of LMs, ignoring the important case of ignoring long-tail entities. In this work, we present a novel method for LM-based-KB completion that is specifically geared to facts about long-tail entities. The method leverages two different LMs in two stages: for candidate retrieval and for candidate verification and disambiguation. To evaluate our method and various baselines, we introduce a novel dataset, called MALT, rooted

in Wikidata. Our method outperforms all baselines in F1, with major gains especially in recall. Chapter 7 is based on the paper [36]:

> *Lihu Chen, Gaël Varoquaux, Fabian Suchanek, Simon Razniewski and Gerhard Weikum. "**Knowledge Base Completion for Long-Tail Entities**." ACL 2023 MATCHING workshop*

I have also been involved in one collaboration during my thesis project which is not included in this thesis:

> *Yaru Wu, Lihu Chen, Benjamin Elie, Fabian Suchanek, Ioana Vasilescu and Lori Lamel. "**Who's speaking? Predicting speaker profession from speech.**." (Full Paper) Under Submission at ICPhS 2023*

2 Preliminaries

In this chapter, we present the basic background knowledge upon which the following chapters are based. We first explain the concept of knowledge bases and their history in Section 2.1. We then explain the task of entity disambiguation and previous work in Section 2.2. Next, we consider the attention mechanism in Section 2.3.3.1. Finally, the concept of the language model is described in Section 2.3.

2.1 Knowledge Bases

Knowledge bases have gradually become a key asset that can be used for a variety of machine intelligence applications such as question answering [256, 232], recommendation system [305, 4], etc. In this section, we begin with a quick history of knowledge bases. Knowledge bases (KBs) comprise salient information about entities, semantic classes to which entities belong, attributes of entities, and relationships between entities.

The concept of knowledge base can be traced back to pioneering work, the Cyc project [153] and the WordNet project [180]. However, these KBs are constructed manually and therefore limited to scope and scale. Later, automatic knowledge acquisition from web and text resources becomes a major research approach, and there are many publicly accessible and large-scale KBs such as KnowItAll [72], DBpedia [9], YAGO [254], Freebase [21], NELL [32], BabelNet [189], ConceptNet [251], Wikidata [274], and DeepDive [243]. These large general-purpose KBs play a huge role in practical artificial intelligence applications and have made substantial impact in the research community.

Next, we discuss the knowledge representation that has emerged as a pragmatic consensus in the research community of entity-centric knowledge bases by reusing the definitions of prior work [255, 284]. Concretely, we discuss the three key concepts of KBs: Entities (Section 2.1.1), Classes (Section 2.1.2), and Relations (Section 2.1.3).

2.1.1 Entities

The most basic element of a KB is an *entity*.

> An **entity** is whatever may be an object of thought.

This definition includes people, places, organizations and also creative works (books, films, songs, etc.) An entity can be real (*Steve Jobs*) as well as fictional (*Harry Porter*), and also general concepts such as *love* and *Buddhism*. In other words, KBs model the real world. This means that they select certain entities of interest entities, give them names, and put them into a structure. Thus, a knowledge base is a structured view of a selected part of the world.

To denote an entity unambiguously, we need an identifier that can refer to only a single entity.

> An **identifier** for an entity is a string of characters that uniquely denotes the entity.

The identifier can be a unique name, but can also be specifically introduced keys such as URLs for websites, Google Scholar URLs etc. In the data model of the Semantic Web, the Resource Description Framework (RDF) [51], identifiers always take the form of Unique Resource Identifiers (URIs). Since the form of an identifier is not always well-readable and directly interpretable, we usually need human-readable labels or names. For example, the mythology character, *Paris*, is a nobleman that appears in a number of Greek legends, and Wikidata has an identifier for him https://www.wikidata.org/wiki/Q167646. Meanwhile, Wikidata contains *"Also known as"* labels for him, e.g., *Alexander* and *Alexandros*. When an entity has several labels, they are called **synonyms** or **alias names**. When an entity label appears in a text, we call that appearance a **mention** of that entity. The label that appears in the mention is called the **surface form** of the entity (two different mentions can use the same surface form).

> A **mention** of an entity in a data source (including text) is a string including acronyms and abbreviations) intended to denote an entity.

Ideally, the mention is one of the labels with which the entity is associated, and it can also be another variant that is not in the KB. We refer to them as unlinkable (NIL) mentions, and these surveys introduce solutions to these mentions [238, 240].

2.1.2 Classes

KBs model real-world entities, which can be categorized into specific classes based on their characteristics. For example, *Paris, Beijing, New York* are all cities. We represent this knowledge by organizing entities into **classes** or **types**.

> A **class**, or interchangeably **type**, is a named set of entities that share a common trait. An element of that set is called an **instance** of the class.

In this setting, the following are classes: The class of cities (i.e., a human settlement of notable size), the class of business companies in the U.S., and the rivers in China. Some instances of these classes are, respectively, *Beijing*, *Amazon* and *Yangtze River*. Since everything is an entity, a class is also an entity, which has an identifier and a label as well.

Meanwhile, there is a hierarchical relation between different classes, i.e., Class A is a subclass of class B if A is a subset of B. For example, *city* is a subclass of *place*. By combining these pairwise relations across all classes, we can thus construct a class hierarchy.

> A **taxonomy** is a directed acyclic graph, where the nodes are classes and there is an edge from class X to class Y if X is a direct subclass of Y.

2.1.3 Properties of Entities: Attributes and Relations

Entities have properties such as birthplace, nationality, height, weight, spouse and so on. KBs capture these in the form of mathematical relations:

> A relation for entities $E_1, ..., E_n$ is a subset of the Cartesian product $E_1 \times ... \times E_n$, along with an identifier for the relation.

For example, the Cartesian product of the relation birth can be denoted as:

```
birth ∈ persons × dates × cities
```

This instance of the birth relation is a ternary tuple, that is, it has three arguments: the person entity, the birthdate, and the birthplace. We can state the birthdate and birthplace of Joe Biden in the relational form:

```
<Joe Biden, 20-11-1942, Scranton(Pennsylvania)>
```

In the above example about the birth relation, the birthdate is a particular entity, which is a design choice whether we regard numerical values like dates, heights or amounts as entities. Often, we want to regard them simply as values for which we do not have any additional properties. For example, the height of Joe Biden is 1.83 meters. The case for binary relations with values as second argument largely corresponds to the **attribute** of entities.

The frequent case of binary relations models the relationship between exactly two entities. The advantage of binary relations is that it can express facts in a self-contained way, even if some properties of entities are missing. For example, if we know *Joe Biden*'s birthplace but not his birthdate, and we use the ternary relation to represent this piece of fact. Then, we have to put a null value to the birthdate position, which makes the representation way awkward and complicated. In KBs, the common practice is to avoid null values and prefer binary relations where we can simply have a triple for the known argument (birthplace). Some KBs focus exclusively on so-called **subject-predicate-object** triples, or **SPO** triples. For example, `<Joe Biden, birthplace, Scranton(Pennsylvania)>` is a triple, and the predicate (P) is `birthplace`, `Joe Biden` and `Scranton(Pennsylvania)` are the subject (S) and object (O), respectively.

2.2 Entity Disambiguation

Natural text in the real-world is extremely ambiguous, i.e., a word or phrase can have multiple meaning in a given context. For example, the mention *"Paris"* in the query *"Pairs is the son of king Priam"* can represent names of cities, songs, people (real or fictional), etc. In the context, the mention *"will"* actually means a fictional character in an American soap opera. To map the mention name to the standard entity, we need the the *Entity Disambiguation (ED)*. Formally, the technology of entity disambiguation aims to map mentions in documents to standard entities in a given knowledge base, and you can reference these surveys [241, 238, 240].

In general, entity disambiguation consists of two key steps: *candidate generation* and *ranking*. Considering that the scale of KBs is large, it is impractical to compare each pair of mention and entity. A trade-off is to construct a manageable set of candidates that are related to the given mention. Then, we rank all candidates by computing the semantic similarity. The candidate entity with the highest score is regarded as the correct mapping. However, some mentions do not have corresponding profiles in a knowledge base due to incompleteness. Therefore, we have to introduce another step for the prediction of *unlinkable mentions*. This part is organized as follows. We present the techniques of candidate generation, ranking and unlinkable mention prediction in Section 2.2.1, 2.2.2 and 2.2.3, respectively. Moreover, we present the newly proposed generative entity disambiguation in Section 2.2.4.

2.2.1 Candidate Generation

Given a mention, this step aims to generate a set of candidate entities from a KB, such as "Paris", to provide a list of entities that are related to the ambiguous mention. Commonly used methods are (1) surface form matching, (2) alias dictionary and (3) methods based on search engines. In the first approach, a candidate list is constructed by matching various surface forms of mentions in the document [185, 149]. There are many heuristics for the generation of mention forms and matching criteria like the Hamming distance [64], edit distance [315], and n-grams. For the second approach, the candidate set can be retrieved by using disambiguation/redirect pages of Wikipedia [73, 318], alias/synonym dictionary of KBs [311]. In the third approach, we can leverage the web search engines to obtain relevant entity pages, e.g., Wikipedia pages, when you query it based on keyword matching [184, 64].

2.2.2 Ranking

After obtaining a set of candidates, the remaining problem is to rank all the candidates according to different types of signals related to the mention and candidates. There are three commonly used features for ranking [113]: (1) Mention-Entity Popularity (2) Mention-Entity Context Similarity (3) Entity-Entity Coherence.

Mention-Entity Popularity. If an entity is frequently referred by the name of the mention, this entity is a likely candidate. For example, in the sentence "Paris is the capital city of France", the mention"Paris" most likely denotes the famous city `Paris` instead of the character `Paris (mythology)`. We call this feature mention-entity popularity, and it describes how often a mention should be linked to a particular entity. Some approaches apply it to entity disambiguation for better performances [120, 225, 83, 96].

Mention-Entity Context Similarity. Mentions have surrounding text, which can be compared to descriptions of entities such as short paragraphs from Wikipedia or keyphrases derived from such texts. For the input sentence "Paris is the song of King Priam", the context words "King Priam" are cues for the mention "Paris" towards the topic of Greek legends, then we know it more likely represents the mythological character entity `Paris (mythology)`. The mention-entity context similarity measures the textual similarity between the context around the mention and the descriptions associated with the candidate entity. There are several methods for capturing the context similarity such bag-of-words [28, 145, 102, 168] and concept vector [64, 45, 242, 258, 77, 144, 170, 291].

Entity-Entity Coherence. In meaningful texts, different entities do not co-occur uniformly at random. For two entities to co-occur, a semantic relationship should hold between them. The existing KB may have such prior knowledge that can be harnessed. For example, the entity `Paris` and `France` often co-appear together in the same document. We refer to this feature as entity-entity coherence. In general, there is one or a few related topics for a given document, and the topical coherence could be beneficial for collectively diambiguating mentions in the same document. We call this series of methods that use the coherence feature as collective entity disambiguation [90, 318, 299].

Apart from the three features, the entity type information is also an useful feature. Some KBs contain a type hierarchy (or taxonomy) and entities that are instances of types. For example, the entity `Paris` is a place and `Paris (mythology)` is a person (mythological). This example shows that even a coarse-grained entity type is beneficial for distinguishing two mentions. A series of disambiguation systems have incorporated entity type information [258, 98, 193].

2.2.3 Unlinkable Mention Prediction

The target entities of some mentions can be absent in the KBs, e.g., there is no entity in a given KB for a newly published book or a new product on the market. We call these mentions unlinkable mentions, and it is necessary for a system to predict that these mentions should not be mapped to any entities, which is known as the NIL prediction task. There are several ways of predicting unlinkable mentions. If the candidate set is empty for a mention, it is considered unlinkable [246]. Or we can set a threshold for the ranking output, mentions with scores below it are unlinkable mentions [206]. It

is also possible to train a binary classifier to predict whether a mention-entity pair is linkable or not [185].

2.2.4 Generative Entity Disambiguation

Although existing models that use these features are effective enough for disambiguating mentions, they usually learn entity representations for comparing the semantic similarity between a mention and an entity, which consumes a large memory footprint for storing the entire real-world KB, e.g., about 24GB to store 1024-dimensional vectors for all of the 6M Wikipedia pages, and the size linearly grows with respect to new entities. To address the issue, GENRE [57] first proposes to link entities by generating their unique names completely based on the context in an autoregressive way. The parameters of this novel generative model scale linearly with vocabulary size instead of entity count and it achieves significant improvements across a series of entity disambiguation and end-to-end entity linking tasks. This generative mode of entity disambiguation has a great advantage in inference speed, and is a promising research direction. Recently, another novel approach cast entity diambiguation to text extraction [11], which have achieved very promising results.

2.3 Language Models

Language models (LMs) are able to estimate the probability distribution over sequences of words. Specifically, a language model can predict the next word for a given sequence. Given the input sequence "the capital of France is", a language model knows that the word Paris should probably appear in the next position. In general, the input for LMs are a large corpus with natural and unlabeled sentences and the output is a model that can predict the target word given a context. In fact, word embeddings [17, 177, 204] are by-products of language models. LMs are the basis for a wide range NLP tasks [6], including sentiment analysis, question answering, machine comprehension, information retrieval, etc [219, 26].

There are several different approaches to modeling language, which can be roughly categorized into three classes: Statistical Language Models (Section 2.3.1), Neural Language Models (Section 2.3.2) and Pre-trained Language Models (Section 2.3.3).

2.3.1 Statistical Language Models

Traditional statistical language models are probabilistic models of the distribution of words in a language. These models aim to estimate the likelihood of a sequence with multiple n words $(w_1, w_2, ..., w_n)$. However, counting the likelihood of each word that may occur in a given context in a language is obviously difficult to achieve, but it has been empirically observed that satisfactory results can be obtained using contexts as small as 3 words [91]. A simple mathematical formulation of such an n-gram model

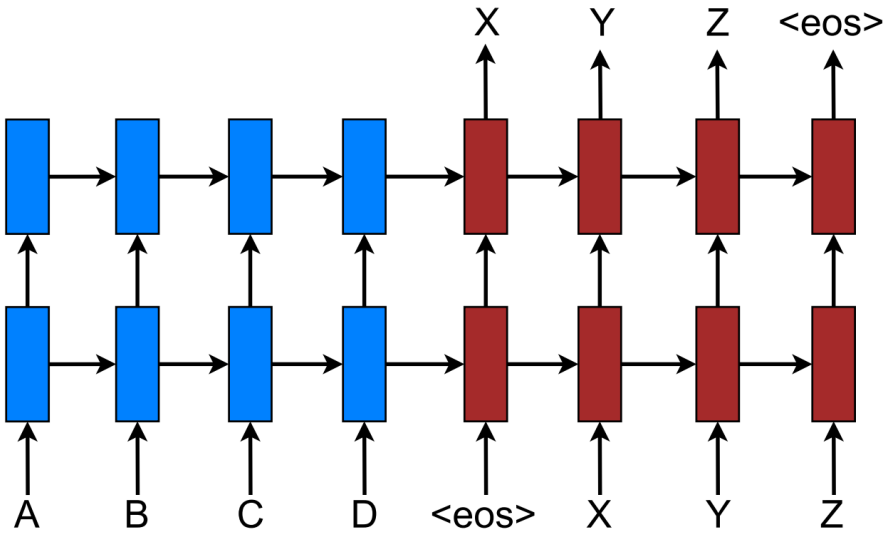

Figure 2.1: An example of neural machine translation [172]: a stacking recurrent architecture for translating a source sentence `A B C D` into a target sentence `X Y Z`. The left blue blocks are the Encoder, and the right red blocks are the Decoder. The arrows show the sequential way of encoding and decoding. Here, `<eos>` marks the end of a sentence.

with a window size equal to T follows:

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t | w_1^{t-1}) \tag{2.1}$$

where w_t is the t-th word and w_i^T refers to the sequence of words from w_i to w_T, i.e., $(w_i, w_{i+1}, ..., w_T)$. $P(w_t|w_1^{t-1})$ refers to the fraction of times w_t appears after the sequence. Actual prediction of the next word given a context is done via maximum likelihood estimation (MLE), over all words in the vocabulary [6].

However, the methods have difficulties in generalizing to word sequences not present in the training set. To address this issue, early attempts use a smoothing strategy, e.g., assuming every new sequence has count one, rather than zero in the training set (this is referred to as *add-one* or *Laplace smoothing*). Another strategy which helps with generalization is the clustering of words in so-called classes [25].

2.3.2 Neural Language Models

Although n-gram LMs with smoothing strategies work out, there are still other problems. A basic problem is the high dimensionality involved in calculating discrete joint distributions of words because the number of parameters grow exponentially by the size of the vocabulary [128]. To mitigate this gap, Neural Network (NN) is introduced for language modeling in continuous space. The first feedforward neural network LM is proposed by Bengio et al. [17], which is able to represent words by using low

dimensional vectors. Some of the notable work includes Word2Vec [177], GloVe [204], fastText [20], etc. Some of the subsequent work uses CNNs or RNNs as the architecture for language models [178, 55].

2.3.3 Pre-trained Language Models

For neural language models, they provide a fixed word embedding for each word. However, words with multiple meanings (Polysemy) are widespread in natural language, e.g., *Paris* can be a city or a mythology character, which means we should understand the word meaning under a particular context. To address this issue, the pre-trained models (or contextualized language models) appeared.

The emergency of Pre-trained Language Models (PLMs) has achieved significant success in the field of Natural Language Processing (NLP) by learning contextual representations on large corpora in a self-supervised manner. ELMo [205] uses a deep bidirectional LSTM model to build word representations, which is able to represent each word according to the entire context in which it is actually used. More specifically, instead of having a table of query word embedding matrices, ELMo is used by feeding the words and their surrounding text into a deep neural network that converts the words into a low-dimensional vector.

Afterwards, transformer-based language models like BERT [59] and GPT [218] have been proposed and soon drastically changed the natural language processing field. Most of these models use the Transformer [270] as the backbone, which is fully attention-based methods. These models adopt a two-step paradigm for solving various NLP tasks: Pre-training and Fine-tuning. For the first step, a large transformer-base model is trained on massive unlabeled corpus by designing unsupervised objectives. For example, BERT is pre-trained by using two unsupervised tasks, Masked Language Model (MLM) and Next Sentence Prediction (NSP). The Masked Language Model task randomly masks some percentage of the input tokens, and then forces the model to predict the masked tokens, e.g., *Paris is the capital of* `[MASK]`. The Next Sentence Prediction task asks the model to predict whether one sentence follows the other. For the second step, the pre-trained model are fine-tuned on limited samples for transferring to the target tasks. Due to the complex pre-training objectives and huge model parameters, large-scale pre-trained language models can effectively capture knowledge from large amounts of unlabeled data. The rich knowledge implicit in the huge parameters can benefit a variety of downstream tasks by storing the knowledge into the huge parameters and fine-tuning it on specific tasks, as has been extensively demonstrated through experimental validation and empirical analysis.

Due to the great success of pre-trained models, the field has become extremely active, with numerous variants emerging, such as RoBERTa [169], ALBERT [147], BART [155], ELECTRA [49], T5 [222]. Recent advances in pre-trained language models have led to the development of powerful language-understanding models, such as GPT-x [219, 26] and ChatGPT, which have resulted in a huge revolution in this field.

The Transformer architecture [270] is widely used by most of pre-trained LMs, and the attention module is the basic component of the Transformer. Next, we first present

the Attention Mechanism 2.3.3.1, then describe the Transformer 2.3.3.2.

2.3.3.1 Attention Mechanism

Attention has arguably become one of the most important concepts in the deep learning field. Many famous models are designed based on the attention mechanism, e.g., Transformer [270] and BERT [59]. In this part, we present the basic principle of the attention, by first introducing the history of attention briefly (Section 2.3.3.1), and then discuss the Normal Attention (Section 2.3.3.1). Finally, we explain the model: Transformer (Section 2.3.3.2).

A Brief History of Attention. Attention is a complex cognitive function that is indispensable to humans [52]. Humans do not process information in its entirety at once. Instead, humans tend to selectively focus on a portion of information when and where it is needed while ignoring other, less important information. For example, when we are driving, drivers pay special attention to moving objects such as pedestrians and other moving cars, and allocate less attention to stationary objects such as billboards on the roadside. This is a way for humans to quickly select high-value information from a large amount of information using limited resources. As mentioned above, the attention mechanism can be used as a resource allocation scheme, which enables humans to focus attention on a certain object consciously and actively [191].

In the field of computers, early research work introduced the attention mechanism into the processing of images [118, 181]. Subsequently, attention mechanisms have been an increasingly common component of neural networks and have been applied to a variety of tasks, such as machine translation [10, 172], text classification [164, 200], recommendation system [279] and so on. Specifically, we illustrate how the attention mechanism works by using an example of machine translation. Figure 2.1 shows a conventional encoder-decoder framework for machine translation. The left blue blocks are the Encoder, which is used to represent the input sentence. The right red blocks are the Decoder, which generates the translated sequence based on the last output of decoder. However, this way is overly dependent on the compression of the entire sentence into a single fixed representation. Actually, the input sequence may contain hundreds of words, therefore, this will inevitably cause information loss and the translation result will not be accurate.

In order to address this issue, we can infuse am attention mechanism into the encoder-decoder model. As shown in Figure 2.2, the attentional model generates a word in a translation at each step of decoding process, it softly searches for a set of positions in the input source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vector associated with these source locations and all previously generated target words. The most important difference between this approach and the basic encoder-decoder is that it does not attempt to encode the entire input sentence as a vector of fixed length. Instead, it encodes the input sentence as a sequence of vectors and adaptively selects a subset of these vectors at the decoding time, which avoid information loss during decoding.

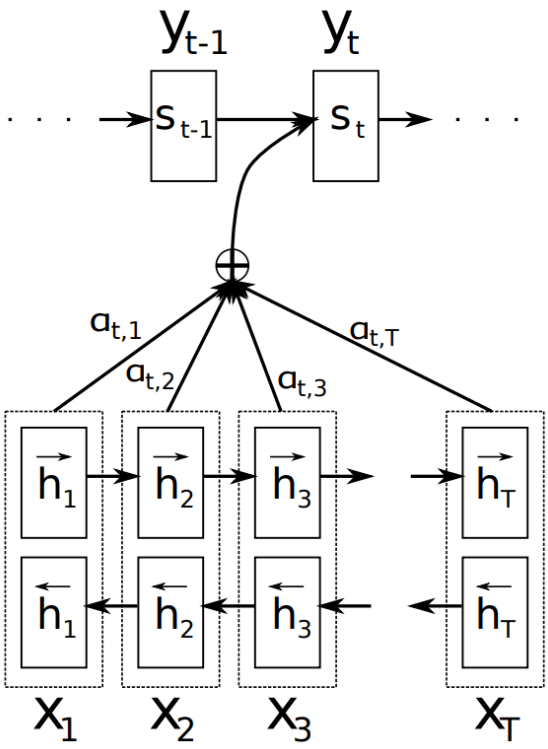

Figure 2.2: The illustration of the attention-based machine translation model [10], which generates the t-th target word y_t given a source sentence $(x_1, x_2, ..., x_T)$.

Normal Attention. The attention mechanism is introduced by Bahdanau et al. [10] to address the bottleneck created by the use of fixed-length vectors, where the decoder has limited access to the information provided by the input, especially considering that the input length can be very long. The attention mechanism can assign different weights to different words at each step of the decoding, so that the input information can be fully leveraged. The normal attention mechanism makes use of three main components, namely the queries, the keys, and the values. In the context of machine translation, each word in the input sentence is given its own query, key and value vectors. Next, the attention mechanism takes a query vector belonging to a specific word in the sequence and computes the correlation between the query and each key. In doing so, it captures the relationship of the word under consideration to other words in the sequence. Then, it adjusts the values based on the attention weights (computed from the correlations) to maintain focus on those words that are relevant to the query. In this way, it generates an attention output for the words by using the attentional weights and value vectors.

Formally, an attention module maps a query and a set of key-value pairs to a new output, where the input query, keys, values, and output are all vectors. The inputs are given as a set of n query vectors, grouped horizontally in a matrix $\mathbf{Q}$. The vectors of keys and the vectors of values, likewise, are stacked as matrices $\mathbf{K}$ and $\mathbf{V}$. Then, the

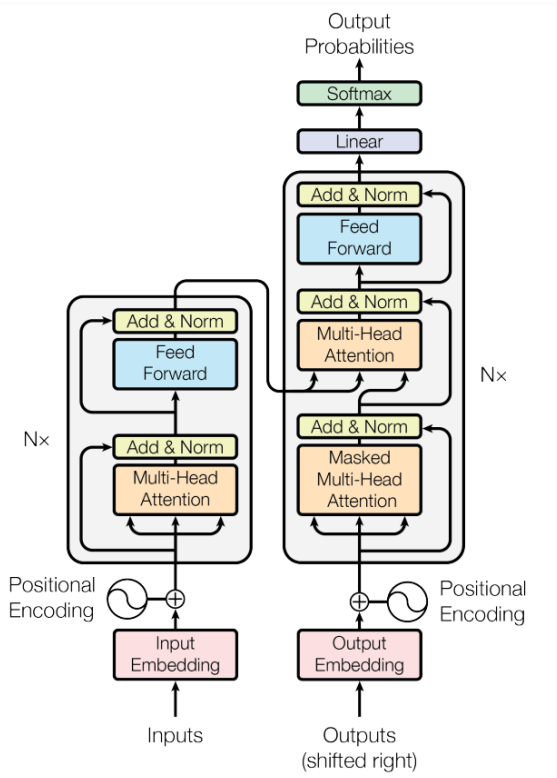

Figure 2.3: The transformer - model architecture [270].

new output matrix $\bar{\mathbf{V}}$ is computed by the Scaled Dot-Product Attention [270]:

$$\bar{\mathbf{V}} = \text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{W}^Q(\mathbf{K}\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}}\right)(\mathbf{V}\mathbf{W}^V) \tag{2.2}$$

Here, $\mathbf{W}^Q \in \mathbb{R}^{d\times d_K}, \mathbf{W}^K \in \mathbb{R}^{d\times d_K}, \mathbf{W}^V \in \mathbb{R}^{d\times d_V}$ are trainable parameter matrices, d is the input dimension of $\mathbf{Q},\mathbf{K},\mathbf{V}$, and d_K, d_V are the dimensions of the Key and Value after projection.

Self-Attention applies attention to a single set of tokens: the key, query, and values are all functions of token embeddings. The input (e.g., a sentence or a sequence of words) is a sequence matrix $\mathbf{X} = \{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n\} \in \mathbb{R}^{n\times d}$, and the output vector of the ith token is given by:

$$\bar{\mathbf{x}}_i = \sum_{j=1}^{n} \frac{\exp(\alpha_{ij})}{Z}\mathbf{x}_j\mathbf{W}^V, \text{where } \alpha_{ij} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}}, Z = \sum_{j=1}^{n}\exp(\alpha_{ij}) \tag{2.3}$$

As tokens appear in the keys and the queries, self-attention can learn the importance of the interactions between words. It has drastically improved performance in many NLP tasks.

2.3.3.2 Transformers

The Transformers rely entirely on self-attention to compute representations of its input and output without using recurrent or convolutional neural networks [270]. The

emergence of the transformer has attracted widespread attention, and various studies have found that it has superior performance in a wide variety of fields [163]. The Transformer is a encoder-decoder model, which is composed of three basic components: input embedding, multi-head self-attention layer and position-wise feed-forward network (FFN) layer. The overall architecture is shown in Figure 2.3.

Input Embedding. Similar to other sequence models, the Transformer uses learned embeddings to transform the input tokens and output tokens into vectors of dimension d_{model} (the width of neural network). The token embedding layer can be considered a lookup table to obtain a representation of the learned vector for each word. Unlike CNNs or RNNs, an attention module cannot intrinsically capture the word order in a sequence. Therefore, the Transformers inject some information about the relative or absolute position of the tokens into the sequence. For this goal, they add **Positional Encodings** to the input embeddings at the bottom of the encoder and decoder blocks. The positional encodings have the same dimension d_{model} as the input embeddings, so that we can add the two. The positional encoding for each position is computed using pre-defined cosine and sin functions:

$$\begin{aligned} PE_{(pos,i)} &= \sin(pos/10000^{2d/d_{model}}) \\ PE_{(pos,2i+1)} &= \cos(pos/10000^{2d/d_{model}}) \end{aligned} \tag{2.4}$$

where pos is the position and i is the dimension. This makes that each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. This design allows the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

Multi-Head Self-Attention. The Transformer projects queries, keys, and values to d_k, d_k, and d_v dimensions with different, learned linear projections to obtain a better representation than a single attention. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. On each of these projected versions of queries, keys and values the transformer then performs the attention function in parallel, yielding multiple dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.4. Multi-head attention linearly projects the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively, which can be denoted as:

$$\begin{aligned} MultiHead(Q,K,V) &= Concat(head_1,...,head_h)W^O \\ \text{where } \ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.5}$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$.

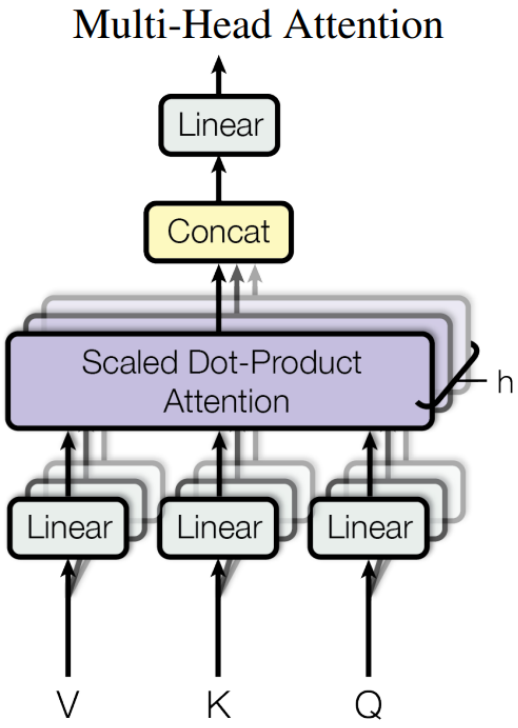

Figure 2.4: An illustration of multi-head self-attention [270]. Multi-Head Attention consists of several attention layers running in parallel.

Feed-Forward Network. Apart from the attention sub-layers, each of the layers in the encoder and decoder has a fully connected feed-forward network, which is applied to each position separately and identically. It consists of two linear transformations with a ReLU activation in between:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.6}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer.

Transformers have achieved significant success in a wide range of artificial intelligence tasks, and a variety of transformer variants have been proposed, such as Linformer [281], Longformer [16], Reformer [141], Shortformer [213] etc.

3

Efficiency: Disambiguating Biomedical Entity with Lightweight Models

Biomedical entity linking aims to map biomedical mentions, such as diseases and drugs, to standard entities in a given knowledge base. The specific challenge in this context is that the same biomedical entity can have a wide range of names, including synonyms, morphological variations, and names with different word orderings. Recently, BERT-based methods have advanced the state-of-the-art by allowing for rich representations of word sequences. However, they often have hundreds of millions of parameters and require heavy computing resources, which limits their applications in resource-limited scenarios. Here, we propose a lightweight neural method for biomedical entity linking, which needs just a fraction of the parameters of a BERT model and much less computing resources. Our method uses a simple alignment layer with attention mechanisms to capture the variations between mention and entity names. Yet, we show that our model is competitive with previous work on standard evaluation benchmarks.

3.1 Introduction

Entity linking (Entity Normalization) is the task of mapping entity mentions in text documents to standard entities in a given knowledge base. For example, the word "Paris" is *ambiguous*: It can refer either to the capital of France or to a hero of Greek mythology. Now given the text "Paris is the son of King Priam", the goal is to determine that, in this sentence, the word refers to the Greek hero, and to link the word to the corresponding entity in a knowledge base such as DBpedia [9] or YAGO [254].

In the biomedical domain, entity linking maps mentions of diseases, drugs, and measures to normalized entities in standard vocabularies. It is an important ingredient for automation in medical practice, research, and public health. Different names of the same entities in Hospital Information Systems seriously hinder the integration and use of medical data. If a medication appears with different names, researchers cannot study its impact, and patients may erroneously be prescribed the same medication twice.

The particular challenge of biomedical entity linking is not the ambiguity: a word

usually refers to only a single entity. Rather, the challenge is that the surface forms vary markedly, due to abbreviations, morphological variations, synonymous words, and different word orderings. For example, *"Diabetes Mellitus, Type 2"* is also written as *"DM2"* and *"lung cancer"* is also known as *"lung neoplasm malignant"*. In fact, the surface forms vary so much that all the possible expressions of an entity cannot be known upfront. This means that standard disambiguation systems cannot be applied in our scenario, because they assume that all forms of an entity are known.

One may think that variation in surface forms is not such a big problem, as long as all variations of an entity are sufficiently close to its canonical form. Yet, this is not the case. For example, the phrase *"decreases in hemoglobin"* could refer to at least 4 different entities in MedDRA, which all look alike: *"changes in hemoglobin"*, *"increase in hematocrit"*, *"haemoglobin decreased"*, and *"decreases in platelets"*. In addition, biomedical entity linking cannot rely on external resources such as alias tables, entity descriptions, or entity co-occurrence, which are often used in classical entity linking settings.

For this reason, entity linking approaches have been developed particularly for biomedical entity linking. Many methods use deep learning: an early work of casts biomedical entity linking as a ranking problem, leveraging convolutional neural networks (CNNs) [159]. More recently, the introduction of BERT has advanced the performance of many NLP tasks, including in the biomedical domain [115, 152, 122]. BERT creates rich pre-trained representations on unlabeled data and achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures. However, considering the number of parameters of pre-trained BERT models, the improvements brought by fine-tuning them come with a heavy computational cost and memory footprint. This is a problem for energy efficiency, for smaller organizations, or in poorer countries.

In this paper, we introduce a very lightweight model that achieves a performance statistically indistinguishable from the state-of-the-art BERT-based models. The central idea is to use an alignment layer with an attention mechanism, which can capture the similarity and difference of corresponding parts between candidate and mention names. Our model is 23x smaller and 6.4x faster than BERT-based models on average; and more than twice smaller and faster than the lightweight BERT models. Yet, as we show, our model achieves comparable performance on all standard benchmarks. Further, we can show that adding more complexity to our model is not necessary: the entity-mention priors, the context around the mention, or the coherence of extracted entities [as used, e.g., in 113] do not improve the results any further. All data and code are available at GitHub [1].

3.2 Related Work

In the biomedical domain, much early research focuses on capturing string similarity of mentions and entity names with rule-based systems [62, 133, 67]. Rule-based systems

[1] https://github.com/tigerchen52/Biomedical-Entity-Linking

are simple and transparent, but researchers need to define rules manually, and these are bound to an application.

To avoid manual rules, machine-learning approaches learn suitable similarity measures between mentions and entity names automatically from training sets [150, 61, 85, 151]. However, one drawback of these methods is that they cannot recognize semantically related words.

Recently, deep learning methods have been successfully applied to different NLP tasks, based on pre-trained word embeddings, such as word2vec [179] and Glove [204]. Other approaches introduce a CNN [159] and RNN [289], respectively, with pre-trained word embeddings, which casts biomedical entity linking into a ranking problem.

However, traditional methods for learning word embeddings allow for only a single context-independent representation of each word. Bidirectional Encoder Representations from Transformers (BERT) address this problem by pre-training deep bidirectional representations from unlabeled text, jointly conditioning on both the left and the right context in all layers. A recent work [122] proposed an biomedical entity normalization architecture by fine-tuning the pre-trained BERT / BioBERT / ClinicalBERT models [59, 115, 152]. Extensive experiments show that their model outperforms previous methods and advanced the state-of-the-art for biomedical entity linking. A shortcoming of BERT is that it needs high-performance machines.

3.3 Our Approach

Formally, our inputs are (1) a *knowledge base* (KB), i.e., a list of entities, each with one or more names, and (2) a *corpus*, i.e., a set of text documents in which certain text spans have been tagged as entity mentions. The goal is to link each entity mention to the correct entity in the KB. To solve this problem, we are given a training set, i.e., a part of the corpus where the entity mentions have been linked already to the correct entities in the KB. Our method proceeds in 3 steps:

Preprocessing. We preprocess all mentions in the corpus and entity names in the KB to bring them to a uniform format.

Candidate Generation. For each mention, we generate a set of candidate entities from the KB.

Ranking Model. For each mention with its candidate entities, we use a ranking model to score each pair of mention and candidate, outputting the top-ranked result.

Let us now describe these steps in detail.

Preprocessing

We preprocess all mentions in the corpus and all entity names in the KB by the following steps:

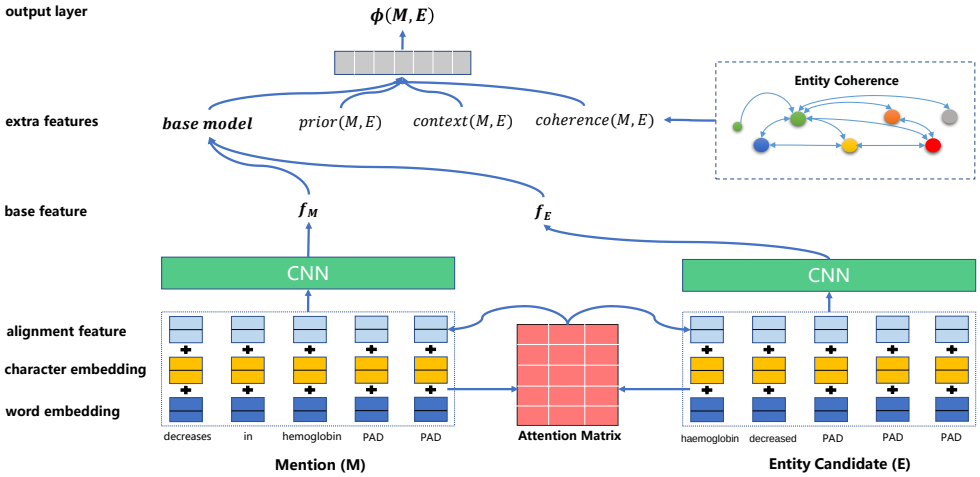

Figure 3.1: The architecture of our ranking model, with the input mention "decreases in hemoglobin" and the input entity candidate "haemoglobin decreased".

Abbreviation Expansion. Like previous work [122], we use the Ab3p Toolkit [250] to expand medical abbreviations. The Ab3p tool outputs a probability for each possible expansion, and we use the most probable expansion. For example, Ab3p knows that "DM" is an abbreviation of "Diabetes Mellitus", and so we replace the abbreviation with its expanded term. We also expand mentions by the first matching one from an abbreviation dictionary constructed by previous work [67], and supplement 20 biomedical abbreviations manually (such as Glycated hemoglobin (HbA1c)). Our dictionary is available in the supplementary material and online.

Numeral Replacement. Entity names may contain numerals in different forms (e.g., Arabic, Roman, spelt out in English, etc.) We replace all forms with spelled-out English numerals. For example, "type II diabetes mellitus" becomes "type two diabetes mellitus". For this purpose, we manually compiled a dictionary of numerals from the corresponding Wikipedia pages. Finally, we remove all punctuation, and convert all words to lowercase.

KB Augmentation. We augment the KB by adding all names from the training set to the corresponding entities. For example, if the training set links the mention "GS" in the corpus to the entity "Adenomatous polyposis coli" in the KB, we add "GS" to the names of that entity in the KB.

Candidate Generation

Our ranking approach is based on a deep learning architecture that can compute a similarity score for each pair of a mention in the corpus and an entity name in the KB. However, it is too slow to apply this model to all combinations of all mentions and all entities. Therefore, we generate, for each mention M in the corpus, a set C_M of candidate entities from the KB. Then we apply the deep learning method only to the

set C_M.

To generate the candidate set C_M, we calculate a score for M and each entity in the KB, and return the top-k entities with the highest score as the candidate set C_M (in our experiments, $k = 20$). As each entity has several names, we calculate the score of M and all names of the entity E, and use the maximum score as the score of M and the entity E.

To compute the score between a mention M and an entity name S, we split each of them into tokens, so that we have $M = \{m_1, m_2, ..., m_{|M|}\}$ and $S = \{s_1, s_2, ..., s_{|S|}\}$.

We represent each token by a vector taken from pre-trained embedding matrix $\mathbf{V} \in \mathbb{R}^{d \times |V|}$ where d is the dimension of word vectors and V is a fixed-sized vocabulary (details in the section of Experimental Settings). To take into account the possibility of different token orderings in M and S, we design the *aligned cosine similarity* (*ACos*), which maps a given token $m_i \in M$ to the most similar token $s_j \in S$ and returns the cosine similarity to that token:

$$ACos(m_i, S) = \max\{cos(m_i, s_j) \mid s_j \in S\} \tag{3.1}$$

The similarity score is then computed as the sum of the aligned cosine similarities. To avoid tending to long text, and to make the metric symmetric, we add the similarity scores in the other direction as well, yielding:

$$sim(M, S) = \frac{1}{|M| + |S|}\left(\sum_{m_i \in M} ACos(m_i, S) + \sum_{s_j \in S} ACos(s_j, M)\right) \tag{3.2}$$

We can now construct the candidate set $C_M = \{\langle E_1, S_1\rangle, \langle E_2, S_2\rangle, ..., \langle E_k, S_k\rangle\}$ where E_i is the id of the entity, and S_i is the chosen name of the entity. This set contains the top-k ranked entity candidates for each mention M. Specifically, if there are candidates whose score is equal to 1 in this set, we will filter out other candidates whose score is less than 1.

Ranking Model

Given a mention M and its candidate set $C_M = \{\langle E_1, S_1\rangle, \langle E_2, S_2\rangle, ..., \langle E_k, S_k\rangle\}$, the ranking model computes a score for each pair of the mention and an entity name candidate S_i. Figure 3.1 shows the corresponding neural network architecture. Let us first describe the base model. This model relies exclusively on the text similarity of mentions and entity names. It ignores the context in which a mention appears, or the prior probability of the target entities. To compute the text similarity, we crafted the neural network following the candidate generation: it determines, for each token in the mention, the most similar token in the entity name, and vice versa. Different from the candidate generation, we also take into account character level information here and use an alignment layer to capture the similarity and difference of correspondences between mention and entity names.

Representation Layer. As mentioned in the , we represent a mention M and an entity name S by the set of the embeddings of its tokens in the vocabulary V. However, not all tokens exist in the vocabulary V. To handle out-of-vocabulary words, we adopt a recurrent Neural Network (RNN) to capture character-level features for each word. This has the additional advantage of learning the morphological variations of words. We use a Bi-directional LSTM (BiLSTM), running a forward and backward LSTM on a character sequence [92]. We concatenate the last output states of these two LSTMs as the character-level representation of a word. To use both word-level and character-level information, we represent each token of a mention or entity name as the concatenation of its embedding in V and its character-level representation.

Alignment Layer. To counter the problem of different word orderings in the mention and the entity name, we want the network to find, for each token in the mention, the most similar token in the entity name. For this purpose, we adapt the attention mechanisms that have been developed for machine comprehension and answer selection [41, 280].

Assume that we have a mention $M = \{\bar{m}_1, \bar{m}_2, ..., \bar{m}_{|M|}\}$ and an entity name $S = \{\bar{s}_1, \bar{s}_2, ..., \bar{s}_{|S|}\}$, which were generated by the Representation Layer. We calculate a $|M| \times |S|$-dimensional weight matrix W, whose element $w_{i,j}$ indicates the similarity between the token i of the mention and the token j of the entity name, $w_{ij} = \bar{m}_i^T \bar{s}_j$. Thus, the i^{th} row in W represents the similarity between the i^{th} token in M and each token in S. We apply a softmax function on each row of W to normalize the values, yielding a matrix W'. We can then compute a vector $\tilde{m}_i$ for the i^{th} token of the mention, which is the sum of the vectors of the tokens of S, weighted by their similarity to $\bar{m}_i$:

$$\tilde{m}_i = \sum_{j=1}^{t} w'_{ij} \bar{s}_j \tag{3.3}$$

This vector "reconstructs" $\bar{m}_i$ by adding up suitable vectors from S, using mainly those vectors of S that are similar to $\bar{m}_i$. If this reconstruction succeeds (i.e., if $\bar{m}_i$ is similar to $\tilde{m}_i$), then S contained tokens which, together, contain the same information as $\bar{m}_i$.

To measure this similarity, we could use a simple dot-product. However, this reduces the similarity to a single scalar value, which erases precious element-wise similarities. Therefore, we use the following two comparison functions[262, 280]:

$$sub(\bar{m}_i, \tilde{m}_i) = (\bar{m}_i - \tilde{m}_i) \odot (\bar{m}_i - \tilde{m}_i) \tag{3.4}$$

$$mul(\bar{m}_i, \tilde{m}_i) = \bar{m}_i \odot \tilde{m}_i \tag{3.5}$$

where the operator $\odot$ means element-wise multiplication. Intuitively, the functions *sub* and *mul* represent subtraction and multiplication, respectively. The function *sub* has similarities to the Euclidean distance, while *mul* has similarities to the cosine similarity – while preserving the element-wise information. Finally, we obtain a new representation of each token i of the mention by concatenating $\bar{m}_i, \tilde{m}_i$ and their

difference and similarity:

$$\hat{m}_i = [\bar{m}_i, \tilde{m}_i, sub(\bar{m}_i, \tilde{m}_i), mul(\bar{m}_i, \tilde{m}_i)] \tag{3.6}$$

By applying the same procedure on the columns of W, we can compute analogously a vector $\tilde{s}_j$ for each token vector s_j of S, and obtain the new representation for the j^{th} token of the entity name as

$$\hat{s}_j = [\bar{s}_j, \tilde{s}_j, sub(\bar{s}_j, \tilde{s}_j), mul(\bar{s}_j, \tilde{s}_j)] \tag{3.7}$$

This representation augments the original representation $\bar{s}_j$ of the token by the "reconstructed" token $\tilde{s}_j$, and by information about how similar $\tilde{s}_j$ is to $\bar{s}_j$.

CNN Layer. We now have rich representations for the mention and the entity name, and we apply a one-layer CNN on the mention $[\hat{m}_1, \hat{m}_2, ..., \hat{m}_{|M|}]$ and the entity name $[\hat{s}_1, \hat{s}_2, ..., \hat{s}_{|S|}]$. We adopt the CNN architecture proposed by [138] to extract n-gram features of each text:

$$f_M = CNN([\hat{m}_1, \hat{m}_2, ..., \hat{m}_M]) \tag{3.8}$$

$$f_E = CNN([\hat{s}_1, \hat{s}_2, ..., \hat{s}_S]) \tag{3.9}$$

We concatenate these to a single vector $f_{out} = [f_M, f_E]$.

Output Layer. We are now ready to compute the final output of our network using a two-layer fully connected neural network:

$$\Phi(M, E) = sigmoid(W_2 \ \ ReLU(W_1 \ f_{out} + b_1) + b_2) \tag{3.10}$$

where W_2 and W_1 are learned weight matrices, and b_1 and b_2 are bias values. This constitutes our base model, which relies solely on string similarity. We will now see how we can add add prior, context, and coherence features.

Extra Features

Mention-Entity Prior. Consider an ambiguous case such as *"You should shower, let water flow over wounds, pat dry with a towel."* appearing in hospital Discharge Instructions. In this context, the disease name *"wounds"* is much more likely to refer to *"surgical wound"* than *"gunshot wound"*. This prior probability is called the *mention-entity prior*. It can be estimated, e.g., by counting in Wikipedia how often a mention is linked to the page of an entity [113]. Unlike DBpedia and YAGO, biomedical knowledge bases generally do not provide links to Wikipedia. Hence, we estimate the mention-entity prior from the training set, as:

$$prior(M, E) = \log count(M, E) \tag{3.11}$$

where $count(M, E)$ is the frequency with which the mention M is linked to the target entity E in the training dataset. To reduce the effect of overly large values, we apply the logarithm. This prior can be added easily to our model by concatenating it in f_{out}:

$$f_{out} = [f_M, f_E, prior(M, E)] \tag{3.12}$$

Context. The context around a mention can provide clues on which candidate entity to choose. We compute a context score that measures how relevant the keywords of the context are to the candidate entity name. We first represent the sentence containing the mention by pre-trained word embeddings. We then run a Bi-directional LSTM on the sentence to get a new representation for each word. In the same way, we apply a Bi-directional LSTM on the entity name tokens to get the entity name representation cxt_E. To select keywords relevant to the entity while ignoring noise words, we adopt an attention strategy to assign a weight for each token in the sentence. Then we use a weighted sum to represent the sentence as cxt_M. The context score is then computed as the cosine similarity between both representations:

$$context(M,E) = \cos(cxt_M, cxt_E) \tag{3.13}$$

As before, we concatenate this score to the vector f_{out}.

Coherence. Certain entities are more likely to occur together in the same document than others, and we can leverage this disposition to help the entity linking. To capture the co-occurrence of entities, we pre-train entity embeddings in such a way that entities that often co-occur together have a similar distributed representation. We train these embeddings with Word2Vec [179] on a collection of PubMed abstracts[2]. Since the entities in this corpus are not linked to our KB, we consider every occurrence of an exact entity name as a mention of that entity.

Given a mention M and a candidate entity E, we compute a coherence score to measure how often the candidate entity co-occurs with the other entities in the document. We first select the mentions around M. For each mention, we use the first entity candidate (as given by the candidate selection). This gives us a set of entities $P_M = \{p_1, p_2, ..., p_k\}$, where each element is a pre-trained entity vector. Finally, the coherence score is computed as:

$$coherence(M,E) = \frac{1}{k}\sum_{i=1}^{k}\cos(p_i, p_E) \tag{3.14}$$

where p_E is the pre-trained vector of the entity candidate E. This score measures how close the candidate entity E is, on average, to the other presumed entities in the document. As before, we concatenate this score to the vector f_{out}. More precisely, we pre-trained separate entity embeddings for the three datasets and used the mean value of all entity embeddings to represent missing entities.

NIL Problem

The NIL problem occurs when a mention does not correspond to any entity in the KB. We adopt a traditional threshold method, which considers a mention unlinkable if its score is less than a threshold τ. This means that we map a mention to the highest-scoring entity if that score exceeds τ, and to NIL otherwise. The threshold τ is learned

[2] ftp://ftp.ncbi.nlm.nih.gov/pubmed/baseline/

from a training set. For datasets that do not contain unlinkable mentions, we set the threshold τ to zero.

Training

For training, we adopt a triplet ranking loss function to make the score of the positive candidates higher than the score of the negative candidates. The objective function is:

$$\theta^* = \arg\min_{\theta} \sum_{D \in \mathscr{D}} \sum_{M \in D} \sum_{E \in C} \max(0, \gamma + \Phi(M, E^+) - \Phi(M, E^-)) \quad (3.15)$$

where θ stands for the parameters of our model. $\mathscr{D}$ is a training set containing a certain number of documents and γ is the parameter of margin. E^+ and E^- represent a positive entity candidate and a negative entity candidate, respectively. Our goal is to find an optimal θ, which makes the score difference between positive and negative entity candidates as large as possible. For this, we need triplets of a mention M, a positive example E^+ and a negative example E^-. The positive example can be obtained from the training set. The negative examples are usually chosen by random sampling from the KB. In our case, we sample the negative example from the candidates that were produced by the candidate generation phase (excluding the correct entity). This choice makes the negative examples very similar to the positive example, and forces the process to learn what distinguishes the positive candidate from the others.

3.4 Experiments

3.4.1 Datasets and Metrics.

We evaluate our model on three datasets (shown in Table 3.1). The **ShARe/CLEF** corpus [211] comprises 199 medical reports for training and 99 for testing. As Table 3.1 shows, 28.2% of the mentions in the training set and 32.7% of the mentions in the test set are unlinkable. The reference knowledge base used here is the SNOMED-CT subset of the UMLS 2012AA [19]. The **NCBI** disease corpus [61] is a collection of 793 PubMed abstracts partitioned into 693 abstracts for training and development and 100 abstracts for testing. We use the July 6, 2012 version of MEDIC [56], which

	ShARe/CLEF		NCBI		ADR	
	train	test	train	test	train	test
documents	199	99	692	100	101	99
mentions	5816	5351	5921	964	7038	6343
NIL	1641	1750	0	0	47	18
concepts	88140		9656		23668	
synonyms	42929		59280		0	

Table 3.1: Dataset Statistics

Model	ShARe/CLEF	NCBI	ADR
DNorm [150]	-	82.20±3.09	-
UWM [85]	89.50±1.02	-	-
Sieve-based Model [67]	90.75±0.96	84.65±3.00	-
TaggerOne [151]	-	88.80±2.59	-
Learning to Rank [296]	-	-	92.05±0.84
CNN-based Ranking [159]	90.30±1.00	86.10±2.79	-
BERT-based Ranking [122]	**91.06±0.96**	89.06±2.63	**93.22±0.79**
Our Base Model	90.10±1.00	89.07±2.63	92.63±0.81
Our Base Model + Extra Features	90.43±0.99	**89.59±2.59**	92.74±0.80

Table 3.2: Performance of different models. Results in gray are not statistically different from the top result.

contains 9,664 disease concepts. The TAC 2017 Adverse Reaction Extraction (**ADR**) dataset consists of a training set of 101 labels and a test set of 99 labels. The mentions have been mapped manually to the MedDRA 18.1 KB, which contains 23,668 unique concepts.

Following previous work, we adopt accuracy to compare the performance of different models.

3.4.2 Experimental Settings

We implemented our model using Keras, and trained our model on a single Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz, using less than 10Gb of memory. Each token is represented by a 200-dimensional word embedding computed on the PubMed and MIMIC-III corpora [307]. As for the character embeddings, we use a random matrix initialized as proposed in this work [105], with a dimension of 128. The dimension of the character LSTM is 64, which yields 128-dimensional character feature vectors. In the CNN layer, the number of feature maps is 32, and the filter windows are $[1,2,3]$. The dimension of the context LSTM and entity embedding is set to 32 and 50 respectively. We adopt a grid search on a hold-out set from training samples to select the value τ, and and find an optimal for $\tau = 0.75$.

During the training phase, we select at most 20 entity candidates per mention, and the parameter of the triplet rank loss is 0.1. For the optimization, we use Adam with a learning rate of 0.0005 and a batch size of 64. To avoid overfitting, we adopt a dropout strategy with a dropout rate of 0.1.

3.4.3 Competitors

We compare our model to the following competitors: **DNorm** [150]; **UWM** [85]; **Sieve-based Model** [67]; **TaggerOne** [151]; a model based on **Learning to Rank** [296]; **CNN-based Ranking** [159]; and **BERT-based Ranking** [122].

Model	ShARe/CLEF	NCBI	ADR
- Character Feature	-1.21	-0.31	-0.30
- Alignment Layer	-3.80	-4.06	-3.17
- CNN Layer	-1.87	-0.93	-0.35
Our Base Method	90.10	89.07	92.63
+ Mention-Entity Prior	+0.33	+0.04	+0.03
+ Context	-0.09	+0.21	-0.24
+ Coherence	-0.02	+0.27	+0.11

Table 3.3: Ablation study

Model	Original ADR	10%	30%	50%	70%	90%
+ Ordering Change	92.63	92.20	92.18	91.95	92.31	92.05
+ Typo	92.63	92.03	91.61	91.38	91.41	91.13

Table 3.4: Performance in the face of typos: Simulated ADR Datasets

3.5 Results

3.5.1 Overall Performance

During the candidate generation, we generate 20 candidates for each mention. The recall of correct entities on the ShARe/CLEF, NCBI, and ADR test datasets is 97.79%, 94.27%, and 96.66% respectively. We thus conclude that our candidate generation does not eliminate too many correct candidates. Table 3.2 shows the performance of our model and the baselines. Besides accuracy, we also compute a binomial confidence interval for each model (at a confidence level of 0.02), based on the total number of mentions and the number of correctly mapped mentions. The best results are shown in bold text, and all performances that are within the error margin of the best-performing model are shown in gray. We first observe that, for each dataset, several methods perform within the margin of the best-performing model. However, only two models are consistently within the margin across all datasets: BERT and our method. Adding extra features (prior, context, coherence) to our base model yields a small increase on the three datasets. However, overall, even our base model achieves a performance that is statistically indistinguishable from the state of the art.

3.5.2 Ablation Study

To understand the effect of each component of our model, we measured the performance of our model when individual components are removed or added. The results of this ablation study on all three datasets are shown in Table 3.3. The gray row is the accuracy of our base model. The removal of the components of the base model is shown above the gray line; the addition of extra features (see the section of) below. If we remove the

Model	Parameters	ShARe/CLEF		NCBI		ADR		Avg	Speedup
		CPU	GPU	CPU	GPU	CPU	GPU		
BERT (large)	340M	2230s	1551s	353s	285s	2736s	1968s	1521s	12.3x
BERT (base)	110M	1847s	446s	443s	83s	1666s	605s	848s	6.4x
TinyBERT$_6$	67M	1618s	255s	344s	42s	2192s	322s	796s	6.0x
MobileBERT (base)	25.3M	1202s	330s	322s	58s	1562s	419s	649s	4.7x
ALBERT (base)	12M	836s	**129s**	101s	24s	1192s	170s	409s	2.6x
Our Base Model	4.6M	**181s**	131s	**38s**	**22s**	**196s**	**116s**	**114s**	-

Table 3.5: Number of model parameters and observed inference time

Alignment Layer (underlined), the accuracy drops the most, with up to 4.06 percentage points. This indicates that the alignment layer can effectively capture the similarity of the corresponding parts of mentions and entity names. The CNN Layer extracts the key components of the names, and removing this part causes a drop of up to 1.87 percentage points. The character-level feature captures morphological variations, and removing it results in a decrease of up to 1.21 percentage points. Therefore, we conclude that all components of our base model are necessary.

Let us now turn to the effect of the extra features of our model. The Mention-Entity Prior can bring a small improvement, because it helps with ambiguous mentions, which occupy only a small portion of the dataset. The context feature, likewise, can achieve a small increase on the NCBI dataset. On the other datasets, however, the feature has a negative impact. We believe that this is because the documents in the NCBI datasets are PubMed abstracts, which have more relevant and informative contexts. The documents in the ShARe/CLEF and ADR datasets, in contrast, are more like semi-structured text with a lot of tabular data. Thus, the context around a mention in these documents is less helpful. The coherence feature brings only slight improvements. This could be because our method of estimating co-occurrence is rather coarse-grained, and the naive string matching we use may generate errors and omissions. In conclusion, the extra features do bring a small improvement, and they are thus an interesting direction of future work. However, our simple base model is fully sufficient to achieve state-of-the-art performance already.

3.5.3 Performance in the Face of Typos

To reveal how our base model works, we further evaluate it on simulated ADR datasets. We generate two simulated datasets by randomly adding typos and changing word orderings of mention names. As described in Table 3.4, as we gradually add typos, the accuracy does not drop too much, and adding 90% of typos only results in a 1.5 percent drop. This shows our model can deal well with morphological variations of biomedical names. Besides, ordering changes almost have no effect on our base model, which means it can capture correspondences between mention and entity names.

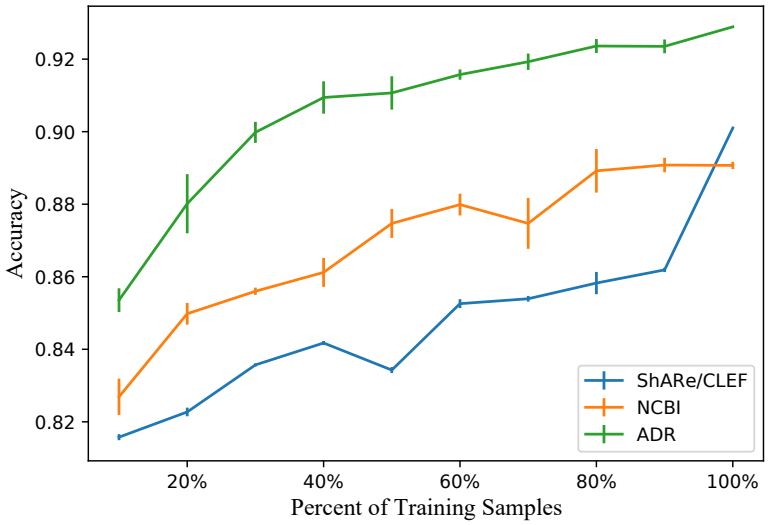

Figure 3.2: Model efficiency on a small amount of data.

3.5.4 Parameters and Inference Time

To measure the simplicity of our base model, we analyze two dimensions: the number of model parameters and the practical inference time. In Table 3.5, we compare our model with BERT models, including three popular lightweight models: ALBERT[147], TinyBERT[125], and MobileBert[260]. Although ALBERT's size is close to our model, its performance is still 2.2 percentage points lower than the BERT_{BASE} model on average.

The second column in the table shows the number of parameters of different models. Our model uses an average of only 4.6M parameters across the three data sets, which is 1.6x to 72.9x smaller than the other models. The third column to the tenth column show the practical inference time of the models on the CPU and GPU. The CPU is described in the Experimental Settings, and the GPU we used is a single NVIDIA Tesla V100 (32G). Our model is consistently the fastest across all three datasets, both for CPU and GPU (except in the fourth column). On average, our model is 6.4x faster than other BERT models, and our model is much lighter on the CPU.

3.5.5 Model Performance as Data Grows

In this section, we study how our model performs with an increasing amount of training samples, by subsampling the datasets. As shown in Figure 3.2, the performance of our base model keeps growing when we gradually increase the number of training samples. When using 50% of the training samples, the accuracies of ShARe/CLEF, NCBI, and ADR dataset are already 0.8342, 0.8747, and 0.9106, respectively. More data leads to better performance, and thus our model is not limited by its expressivity, even though it is very simple.

3.6 Conclusion

In this paper, we propose a simple and lightweight neural model for biomedical entity linking. Our experimental results on three standard evaluation benchmarks show that the model is very effective, and achieves a performance that is statistically indistinguishable from the state of the art. BERT-based models, e.g., have 23 times more parameters and require 6.4 times more computing time for inference. Future work to improve the architecture can explore *1)* automatically assigning a weight for each word in the mentions and entity names to capture the importance of each word, depending, e.g., on its grammatical role; *2)* Graph Convolutional Networks (GCNs) [140, 290] to capture graph structure across mentions and improve our notion of entity coherence.

4 Generalizability: Disambiguating Acronym in General Domain

Acronym Disambiguation (AD) is crucial for natural language understanding on various sources, including biomedical reports, scientific papers, and search engine queries. However, existing acronym disambiguation benchmarks and tools are limited to specific domains, and the size of prior benchmarks is rather small. To accelerate the research on acronym disambiguation, we construct a new benchmark named GLADIS with three components: (1) a much larger acronym dictionary with 1.5M acronyms and 6.4M long forms; (2) a pre-training corpus with 160 million sentences; (3) three datasets that cover the general, scientific, and biomedical domains. We then pre-train a language model, *AcroBERT*, on our constructed corpus for general acronym disambiguation, and show the challenges and values of our new benchmark.

4.1 Introduction

An acronym is an abbreviation formed from the initial letters of a longer name. For instance, the following two sentences contain the acronym *"AI"*: (1) *This is the product's first true **AI** version, and it understands your voice instantly.* (2) *In the United States, the **AI** for potassium for adults is 4.7 grams.* The long forms (or expanded forms) for the same acronym are *"Artificial Intelligence"* and *"Adequate Intake"*, respectively.

Acronym Disambiguation (AD) is the task of mapping a given acronym in a given sentence to the intended long form. Acronym disambiguation is crucial for downstream tasks such as information extraction, machine translation, and query analysis in search engines [119, 117]. Acronym disambiguation is also important for humans: acronyms may make a text more difficult to understand for readers who are not familiar with the specific domain. A study on a Microsoft question answering forum found that only 7% of the acronyms co-occur with their corresponding long forms, which confuses the readers about the meaning of a text [161].

Acronym Disambiguation has received more attention in the past few years. The first step in acronym disambiguation is usually the creation of a dictionary, i.e., a mapping of each acronym to one or more long forms. Early systems extracted acronyms

ID	Long Form	Popularity	Domain
1	***Artificial Intelligence***	★★★★★	Computer Science
2	*Adequate Intake*	★★★★	Food and Nutrition
3	*Aromatase Inhibitor*	★★★	Chemistry
4	*Apoptotic Index*	★★★	Biomedicine
5	*Asynchronous Irregular*	★★★	Neuroscience
6	*Amnesty International*	★★	Organization
7	*Anterior Insula*	★★	Biomedicine
8	*Air India*	★★	Organization
9	***Article Influence***	★★	Science
.....			
2243	*Agricultural Implement*	★	Agriculture

Table 4.1: Long form candidates for the acronym *"AI"* from our acronym dictionary. The SciAD benchmark [272] only includes two long terms (black) in the scientific domain. The popularity is the occurrence frequency in our collected corpora.

and their definitions automatically from texts by rule-based [237] or supervised [188] methods. Once a dictionary is available, acronym disambiguation methods expand acronyms in a given text by capturing the contexts for specific domains, e.g., the enterprise domain [161], biomedical texts [127], and scientific papers [35]. Madog [271] was the first general and web-based system, recognizing and expanding acronyms across multiple domains. Several benchmarks have also been constructed, including for the biomedical area [261] and the scientific area [SciAD, 272]. Several methods fine-tuned SciBERT [15] on SciAD to disambiguate acronyms in scientific documents [197, 312, 156].

Although these works have significantly advanced the progress of acronym disambiguation, they suffer from three main limitations. First, most existing dictionaries (and benchmarks) focus on one specific domain. In real-world applications, however, the input text may be general, cross-domain, or of an unspecified domain (as in search engine queries). Second, existing dictionaries are limited in size. For example, there are only two long forms for the acronym *"AI"* in SciAD (Table 4.1), which is constructed from arXiv. However, we find that the two long forms *"Asynchronous Irregular"* and *"Anterior Insula"* also appear in scientific papers on arXiv [89, 269], and the acronym *"AI"* also appears separately without the long form in sentences. In our work, we actually find at least 2 243 different long forms for *"AI"*. Besides, SciAD suffers from the problem of data leakage, because the train and test sets have overlapping pairs of acronym and long form. Finally, current general AD systems such as MadDog [271] rely on static word embeddings and LSTMs (Long Short Term Memory [112]). Thus, they do not leverage pre-training on large corpora, which drives the current state of the art in most NLP tasks with contextual embeddings like BERT [59].

With this work, we aim to improve Acronym Disambiguation along two dimensions: First, we automatically construct GLADIS, a **G**eneral and **L**arge **A**cronym

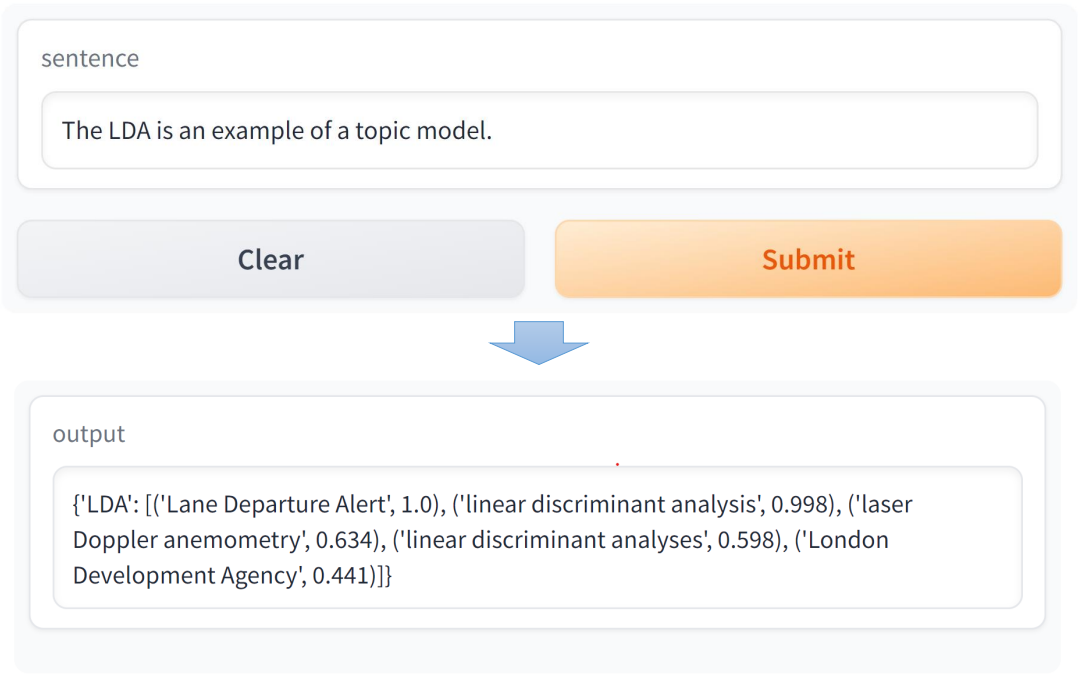

Figure 4.1: An end-to-end acronym disambiguation system using AcroBERT. You can try the Demo at https://huggingface.co/spaces/Lihuchen/AcroBERT.

DISambiguation benchmark that includes a larger dictionary, a pre-training corpus and three datasets covering the general, biomedical, and scientific domains. Our dictionary contains 1.5M acronyms and 6.4M long forms, which trumps existing dictionaries by a factor of 3. We complement this dictionary by three domain-specific datasets for acronym disambiguation, which are adapted from three existing human-annotated and crowd-sourced datasets [182, 193, 272]. The pre-training corpus has 160 million sentences with acronyms, collected from the Pile dataset [81] with a rule-based algorithm [237]. Second, we propose AcroBERT, the first pre-trained language model for general acronym disambiguation. Our experiments show that this model outperforms existing systems across multiple domains. All code and data are publicly available at https://github.com/tigerchen52/GLADIS.

4.2 Related Work

4.2.1 Acronym Identification and Disambiguation

To expand acronyms, there are usually two sub-tasks: Acronym Identification (AI), which creates a dictionary of acronyms and their definitions from a given document, and Acronym Disambiguation (AD), which aims to link acronyms in the input text to the correct long forms from a dictionary.

The study of acronym identification has a long history. Early work observed that acronyms and their long forms appear frequently together in a document, as

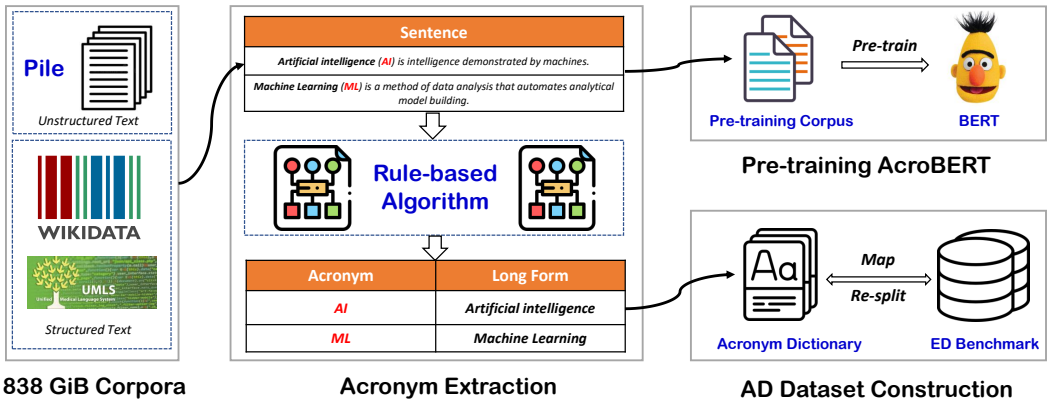

Figure 4.2: Framework of our benchmark construction. The *"ED"* in the lower right corner means *"Entity Disambiguation"*.

in *"Artificial Intelligence (AI)"*. Based on this pattern, many approaches identify and extract acronyms by using rules [302, 148, 216, 201, 303, 237, 2, 8, 192, 250, 271] or supervised methods [34, 188, 146, 186, 166, 292, 314]. In our work, we build on previous work [237] for Acronym Identification, and focus mainly on disambiguation.

As for acronym disambiguation, early solutions manually designed features to score each pair of acronyms and long forms, by either unsupervised [119, 109] or supervised machine learning [196, 304, 252, 76, 161]. Later, deep learning approaches were introduced to the task, using embeddings to represent word sequences. The methods can be categorized as static embedding-based [294, 158, 35] and dynamic embedding-based [127, 197, 312, 156], where the former generates fixed representations for words in a pre-defined vocabulary and the latter can represent arbitrary words dynamically based on specific contexts. One main limitation of these methods is that they are domain-specific systems that can be applied only to a certain field such as the biomedical domain or scientific documents. To generalize the system, Abbreviation Expander [48] and MadDog [271] are proposed, both of which can be used in multiple domains. In this paper, we improve over the performance of these systems by adapting transformer-based methods and pre-training strategies.

4.2.2 Existing benchmarks

Most current public datasets for acronym expansion are focused on a particular domain, such as the biomedical domain [261, 285] or science [35, 272]. Some works adopt two domain-specific datasets for better evaluations [47, 273]. The main limitation of these benchmarks is two-fold: first, their acronym dictionaries are rather small. For instance, the average number of candidates per acronym in the SciAD benchmark [272] is 3.15 while in our benchmark the number is greater than 200. Second, there are no AD evaluation sets that cover multiple domains. We also note that, in SciAD, the train and test sets have overlapping pairs of acronym and long form. For example, the pair $\langle$*CT, Computed Tomography*$\rangle$ appears in the training, validation, and test sets.

4.3 Constructing GLADIS

Our GLADIS benchmark consists of three components: a dictionary, a pre-training corpus, and three domain-specific datasets.

4.3.1 Dictionary and Pre-training Corpus

We propose an acronym dictionary that addresses the shortcomings of existing dictionaries (Section 4.2.2) by being (1) cross-domain and (2) large in size. To construct this dictionary, we apply rule-based extraction on a large set of corpora that contain acronym definitions. In this process, we can also obtain a large number of sentences containing acronyms as the pre-training corpus.

Subset	Domain	Size (GiB)
Pile-CC	Web Archive files	227.12
Books3	Books	100.96
Github	Open-source codes	95.16
PubMed Central	Biomedical articles	90.27
OpenWebText2	Reddit submissions	62.77
ArXiv	Research papers	56.21
FreeLaw	Legal proceedings	51.15
Stack Exchange	Question-answer texts	32.20
USPTO Backgrounds	Patents	22.90
PubMed Abstracts	Biomedical abstracts	19.26
OpenSubtitles	Subtitles	12.98
Gutenberg (PG-19)	Western literatures	10.88
DM Mathematics	mathematical problems	7.75
Wikipedia (en)	Wikipedia pages	6.38
BookCorpus2	Books	6.30
Ubuntu IRC	Chatlog data	5.52
EuroParl	Proceedings	4.59
HackerNews	Comments of social news	3.90
YoutubeSubtitles	YouTube subtitles	3.73
PhilPapers	Philosophy publications	2.38
NIH ExPorter	Awarded applications	1.89
Enron Emails	Emails	0.88
Wikidata Alias	Alias Table	11.00
UMLS Concept	Biomedical Vocabulary	1.96
Total	-	838.14

Table 4.2: Sources for acronym extraction. All corpora except Wikidata Alias and UMLS Concept are from Pile [81].

Input Corpora. For the textual data source, we use the Pile dataset [81], an 825 GiB English corpus constructed from 22 diverse high-quality subsets. Pile [81] is an 825 GiB English text corpus designed to train large-scale language models, which is

constructed from 22 diverse high-quality academic or professional sources. Pile is constructed from existing or newly introduced datasets, and we present these sources here. **Pile-CC** is a collection of web pages, metadata and texts, which is extracted from jusText [70]. **Books3** is a book dataset of fiction and nonfiction books derived from Bibliotik [1]. **Project Gutenberg** has classic Western literature derived from PG-19 [220]. **OpenSubtitles** provides a large corpus of English subtitles from movies and television shows collected by this work [265]. **DeepMind Mathematics** is a collection of many different types of mathematics questions [233]. **BookCorpus2** is an expanded version of BookCorpus [317], a corpus of books from the web. **EuroParl** is a corpus of parallel text in 11 languages from the proceedings of the European Parliament[143]. **Enron Emails** is a large set of email messages, which contains 619,446 messages belonging to 158 users [142].

We also make use of structured knowledge from knowledge bases, namely the Alias Table from Wikidata and the Concept Names from UMLS. Both of them contain alternate names for canonical entities, and these may be acronyms or not. To consider only the acronyms, we produce pairs of the canonical name and an alternate name in the form *"canonical form (alternate name)"*. The rule-based algorithm will then decide whether to extract an acronym or not. Table 4.2 shows the statistics of our sources. They cover a wide range of domains including Web pages, books, scientific and biomedical papers, legal documents, etc.

Acronym Extraction. To extract acronyms from the textual sources, we use a rule-based algorithm [237]. It assumes that acronyms follow a predictable pattern, e.g., *long form (acronym)* or *acronym (long form)*, and then uses rules to extract candidate pairs by identifying parentheses and surrounding tokens. Experimental results show that this simple algorithm achieves 95% precision and 82% recall, averaged over two datasets. As the method has good results at low time complexity, we decided to not adopt more sophisticated methods. Some extracted samples are shown in Table 4.3 in the appendix. A manual evaluation on a random sample of 100 extracted sentences yields a precision of 94%.

Dictionary Construction. Next, we build a large-scale acronym dictionary with frequencies (popularity) by merging the extracted outputs. This merger may regroup duplicate long forms for an acronym, e.g., *"convolutional neural network"*, *"convolutional-neural network"* or *"convolutional neural networks"*. Therefore, we merge long forms that are identical after stemming and removing punctuation. In our case, the above three forms are merged into *"convolutional neural network"*. We keep the most frequent, unpreprocessed, long form as the canonical name in our dictionary, discarding other forms. There are still some noisy long forms that cannot be merged, caused by typos and nested acronyms. However, a manual evaluation on a sample shows that 94% of the long forms are clean. If the long forms are weighted by their frequency, the percentage of clean forms increases to 97%. Most notably, all most frequent long forms for a given acronym were clean in our sample. The statistics of our dataset are shown in Table 4.4. Our resource will be the largest publicly available dictionary for acronyms that covers various domains.

[1] https://twitter.com/theshawwn/status/1320282149329784833

Acronym	Long Form	Provenance
ELEC	*Election Law Enforcement Commission*	*Christie, some legislators and the state Election Law Enforcement Commission (**ELEC**), have joined the comptroller in voicing support for the elimination of the loophole.*
ISR	*in-stent restenosis*	*Although conventional stents are routinely used in clinical procedures, clinical data shows that these stents are not capable of completely preventing in-stent restenosis (**ISR**) or restenosis caused by intimal hyperplasia.*
IL-6	*interleukin-6*	*Consistent blood markers in afflicted patients are normal to low white cell counts and elevated interleukin-6 (**IL-6**) levels which, among its many activities, signal the liver to increase synthesis and secretion of CRP.*
PCP	*Planar cell polarity*	*Establishment of photoreceptor cell polarity and ciliogenesis Planar cell polarity (**PCP**)-associated Prickle genes (Pk1 and Pk2) are tissue polarity genes necessary for the establishment of PCP in Drosophila.*
DEP	*dielectrophoretic*	*They included: a particle counter, trypan blue exclusion (Cedex), an in situ bulk capacitance probe, an off-line fluorescent flow cytometer, and a prototype dielectrophoretic (**DEP**) cytometer.*
AQP3	*aquaporin3*	*The laxative effect of bisacodyl is attributable to decreased aquaporin-3 expression in the colon induced by increased PGE2 secretion from macrophages.The purpose of this study was to investigate the role of aquaporin3 (**AQP3**) in the colon in the laxative effect of bisacodyl.*

Table 4.3: Samples of extracted acronyms, long forms and provenances by using the rule-based algorithm from this work [237].

	Short Form	Long Form	Avg
SciAD [272]	732	2,308	3.15
MadDog [271]	426,389	3,781,739	8.87
Ours	1,542,819	6,381,257	4.14

Table 4.4: Statistics for three acronym dictionaries. The "Avg" column shows the average number of long forms per acronym.

Pre-training Corpus. While building the dictionary, we can also collect the sentences that contain acronyms for pre-training. For example, the following sentence contains the acronym *ELEC*: *"Christie, some legislators and the state Election Law Enforcement Commission (**ELEC**), have joined the comptroller in voicing support for the elimination of the loophole."* For pre-training, the long form *Election Law Enforcement Commission* is removed, and we then force the model to restore the long form from our constructed dictionary, based on the input sentence and the acronym. In total, we collect a pre-training corpus with ˜160 million sentences. More examples are shown in Table 4.3.

	train	valid	test	unique short form	long forms per acronym	overshadowed ratio
General	13,269	7,024	7,125	1,147	248	29.8%
Scientific	28,023	14,134	14,066	2,922	262	68.7%
Biomedical	6,295	3,150	3,149	2,909	278	27.4%

Table 4.5: Statistics of our new Acronym Disambiguation Benchmark. The last column shows the ratio of overshadowed samples in the dataset: long forms with the same acronym but not the most popular one.

4.3.2 Acronym Disambiguation Dataset

We use our acronym dictionary to construct new, larger datasets for evaluating AD systems. To automatically construct the datasets, we adapt the existing two Entity Disambiguation datasets by replacing the long form of entity with the acronym. For example, one sentence in Medmentions [182] contains the long form of Cerebral Blood Flow: *"The reconstructed volume was then compared with corresponding magnetic resonance images demonstrating that the volume of reduced **Cerebral Blood Flow** agrees with the infarct zone at twenty-four hours"*. The dataset provides the unique ID of this long form in UMLS (*C1623258*), and we use it to find the acronym *CBF* in UMLS. Therefore, a new sample can be obtained by replacing the long form with its corresponding acronym.

Specifically, we use the following human-annotated and crowd-sourced datasets: **WikilinksNED Unseen Mentions** [193] is an Entity Disambiguation dataset, i.e., a set of text documents that have mentions of entities, together with a reference knowledge base (KB) that contains, for each entity, one or several names. WikilinksNED Unseen

Mentions re-splits the WikilinksNED dataset [71] to ensure that all mentions in the validation and test sets do not appear in the training set. This is a large-scale, crowd-sourced ED dataset from websites in various fields, which is significantly noisier and more challenging than prior datasets. The reference KB is Wikidata (or Wikipedia), and we adapt this WikilinksNED Unseen Mentions to an AD dataset in the **general** domain.

Medmentions [182] is an entity disambiguation dataset of 4,392 PubMed papers that were annotated by professional and experienced annotators in the biomedical domain. The reference KB is UMLS [19], and this is a **biomedical** dataset.

SciAD [272] is the previously mentioned acronym disambiguation dataset in the **scientific** domain.

SciAD is already an AD dataset, and we only re-split it to avoid data leakage. As for the two ED datasets, they both provide a unique ID to the reference KB for each long form. We then replace the long forms with the acronyms from their corresponding reference KB, i.e., Wikidata and UMLS. To make sure this replacement is correct, we apply the rule-based algorithm [237] to the pair of long form and acronym again for verification. We manually checked 100 random sentences constructed in this way and did not find problematic cases. Hence, this semi-synthetic construction results in a dataset of natural text in which the long form and the acronym are mutually replaceable in the context. Besides, the pair is added to our dictionary with a frequency of 1 if it does not appear in our dictionary. For the WikilinksNED dataset, we use the taxonomy of YAGO 4 [203] to label each long form with a top-level class. For example, *"rhythm and blues"* is a `CreativeWork` and *"United States Navy"* is an `Organization`.

We then partition the three datasets separately into training, test, and validation set, ensuring that the acronyms in the training set do not appear in the validation and test sets. We repartition the datasets at the ratio of 6:2:2. Table 4.5 gives the statistics of this new benchmark. It is not only larger but also more challenging than existing benchmarks, because acronyms in our benchmark have more than 200 candidates on average. Moreover, it contains many *overshadowed forms* [214], which means that an acronym has to be disambiguated to a long form that is not the most popular long form for that acronym. For example, *"Adequate Intake"* is overshadowed by the more popular form *"Artificial Intelligence"* for the acronym *"AI"*.

4.4 AcroBERT

We can now capitalize on our dictionary and pre-training corpus to propose a new method for acronym disambiguation. It takes as input (1) a dictionary of acronyms with their long form(s), and (2) a large-scale corpus that contains acronyms (we assume that the boundaries of the acronym have already been recognized). Our goal is to pre-train a language model for acronym disambiguation, which has a strong generalizability across multiple domains.

The Pre-training Strategy of BERT. We adapt the BERT model for our purpose. BERT is pre-trained by using two unsupervised tasks, Masked Language Model (MLM)

and Next Sentence Prediction (NSP). The Masked Language Model task randomly masks some percentage of the input tokens, and then forces the model to predict the masked tokens, similar to a cloze task. The Next Sentence Prediction task asks the model to predict whether one sentence follows the other.

The Next Sentence Prediction task can be used to predict, from the input text (e.g., *"This is the product's first true **AI** version, and it understands your voice instantly."*), the correct long form (*"Artificial Intelligence"*). Here, the model learns to judge whether the input context that contains the acronym *"AI"* is coherent with the long form *"Artificial Intelligence"*. The Masked Language Model task can memorize the correlation of tokens between the context sequence and long form. Thus, the model learns that the phrase *"Artificial Intelligence"* often co-occurs with *"product"* or *"understand"*.

However, we find that this naive technique does not perform well (see the ablation studies in Table 3.3). We believe that the reason is that the acronym is usually ambiguous with many candidates (as shown in Table 4.1), so that the model has difficulties predicting the correct long form by only using the cross-entropy loss of the binary classification. We also observe that the Masked Language Model loss is so small that the model focuses on adapting the Next Sentence Prediction task only.

AcroBERT. To mitigate the weaknesses of the original BERT, we pre-train an adapted BERT, called AcroBERT, by slightly adjusting the Next Sentence Prediction task. The framework is shown in Figure 4.3. It aims to bring the positive sample pairs closer together, and to push apart the negative sample pairs. We find that already such a simple model can perform very well. For each pair of a candidate long form and a sentence with an acronym, we compose an input for the Next Sentence Prediction task as *"`[CLS]` long form `[SEP]` sentence `[SEP]`"*. Then, we obtain representations of this sequence by applying BERT to this new input. The final hidden vector $\mathbf{e}^{\texttt{[CLS]}} \in \mathbb{R}^H$ of the first input token (`[CLS]`) is used as the aggregate representation, where H is the dimension of the hidden vector. Next, the scores for the binary classification are:

$$\mathrm{P}(y) = \mathrm{softmax}(\mathbf{e}^{\texttt{[CLS]}}\mathbf{W}), y \in \{0,1\} \tag{4.1}$$

where $\mathbf{W} \in \mathbb{R}^{H\times 2}$ is a trainable matrix initialized with the weights of the original BERT, and the label 0 signifies that this pair of sentences are coherent. We use $d = \mathrm{P}(y=1)$ as the distance between the candidate and the context, and we want the distances of negative pairs to be larger than for positive pairs. For this, we use a triplet loss function that aims to assign higher scores to the correct candidates that match the topic of the input sentence while reducing the scores of irrelevant candidates:

$$\mathscr{L} = \max\left\{0, \lambda - d_{\text{neg}} + d_{\text{pos}}\right\} \tag{4.2}$$

where λ is the margin value, and d_{pos} and d_{neg} are the distances for positive and negative pairs, respectively.

The negatives in this triplet framework can be randomly sampled from the dictionary. However, we observe that such random negatives contribute less to the training and

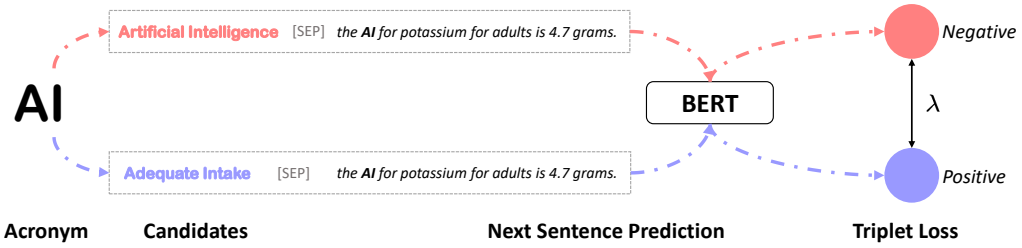

Figure 4.3: The pre-training strategy of AcroBERT. λ is a margin between positive and negative pairs, here $\langle$*Adequate Intake, AI*$\rangle$ and $\langle$*Artificial Intelligence, AI*$\rangle$.

result in slower convergence because the initial model can easily distinguish these triplets. Therefore, it is crucial to select harder triplets that are active and beneficial to the training. For this purpose, we introduce a certain number of ambiguous negatives to each mini-batch, e.g., *"Artificial Intelligence"* can be added to the input context as an ambiguous negative sample for the positive pair *"Adequate Intake* `[SEP]` *In the United States, the* ***AI*** *for potassium for adults is 4.7 grams."* Through the pre-training step, AcroBERT is able to identify the correct long form with the most consistent theme from numerous candidates based on the input context.

4.5 Experiments

In this section, we compare AcroBERT empirically to other acronym disambiguation approaches.

4.5.1 Experimental Settings

Datasets. We use the following datasets for evaluation: Our GLADIS benchmark consists of three subsets covering the General, Scientific, and Biomedical domains. This benchmark is more challenging than prior work due to a large number of ambiguous long forms: each acronym has around 200 candidates on average. We also evaluate AcroBERT on two existing datasets: UAD [47] and SciAd [272]. They are general and scientific AD datasets, respectively. We reuse the test set of Medmentions but use the UMLS as the target dictionary. We refer to them as **datasets with fewer candidates** because they have fewer candidates per acronym.

Our GLADIS benchmark consists of three subsets covering the General, Scientific, and Biomedical domains. It is a very challenging benchmark, due to a large number of ambiguous long forms, as described in Section 4.3.2.

- **General** has 45K samples gathered from the WikilinksNED Unseen Mentions [193].
- **Scientific** is adapted from SciAD [272] with 56K samples, and the long forms in the original dataset are mapped to the new acronym dictionary. We re-split the training, validation and test sets to assure there are no overlaps.

- **Biomedical** includes 12K samples obtained from Medmentions [182].

Datasets with fewer candidates per acronym.

- **UAD** [47] is gathered from the English Wikipedia and we use the manually labeled 7K samples for evaluation.
- **SciAD** [272] is a human-annotated dataset for the scientific domain with 62K samples gathered from the ArXiv preprint papers, and the validation set with 6K samples is used for experiments.
- **Biomedical-UMLS** is a dataset with 3K samples obtained from the test set in our benchmark by using the UMLS concepts as the acronym dictionary

The average candidates per acronym for the three datasets are 2.1, 3.1, and 34.2, respectively.

4.5.2 Competitors

We compare our approach to the following publicly available competitors:

- BM25 [228] is a classical ranking function in information retrieval.
- Popularity-Ours is a baseline that uses the frequency of long forms of our collected pre-training datasets.
- BERT [59] is a strong baseline, which pre-trains contextual language models on large corpora. The scores for the NSP task can be used for the acronym disambiguation.
- FastText [20] is a character-level embeddings and can produce representations for arbitrary words. In this experiment, we first represent the input sentence and candidates by the sum of word embeddings from FastText. Then, all candidates are ranked by their cosine similarity score.
- MadDog [271] is a web-based acronym disambiguation system for multiple domains. It first creates chunks in which all samples with the same acronyms are assigned to the same chunks. After, a separate Bi-LSTM model is trained for each chunk. To deploy the MadDog server, it needs at least 125 GB of disk space and 70 GB of RAM memory [2].
- BioBERT [152] is a biomedical language representation model mainly pre-trained on PubMed Abstracts and PMC Full-text articles, which is a strong baseline in the biomedical domain.
- SciBERT [15] is a scientific language model based on BERT pre-trained on a large multi-domain corpus of scientific publications, which can improve performance on downstream scientific NLP tasks.

[2] https://github.com/amirveyseh/MadDog

Besides, we introduce a Popularity-Ours baseline that uses the frequency of long forms of our collected pre-training datasets. We do not compare to general entity linking methods, because prior work has already found that general systems like AIDA [113] tend to lag behind acronym disambiguation models by 10-30 absolute percentage points [161].

Implementation Details. All approaches are implemented with PyTorch [202] and HuggingFace [288]. When pre-training AcroBERT, the model is initialized by the parameters in the original BERT, and then pre-trained on our collected datasets for one epoch. In total, there are ˜160 million samples in this corpus, covering various domains. The batch size is 32, and we use Adam [139] with a learning rate $2e-5$ for optimization. The learning rate is exponentially decayed for every 10,000 steps with a rate of 0.95. The margin of triplet loss is 0.2 and the number of ambiguous negatives is 2 for each mini-batch.

For the fine-tuning stage, each competitor model is initialized with the pre-trained parameters from HuggingFace, and we use AcroBERT after pre-training for comparison. All models are fine-tuned by using the Triplet loss. All parameters of each model are fine-tuned in this experiment, across all domains by using the same hyper-parameters. The batch size is 8 and the learning rate is among $[1e-5, 8e-6, 6e-6, 4e-6, 2e-6]$ for the Adam optimizer. The model that has the best performance on the validation set among 5 epochs is evaluated on the test set. We use one NVIDIA Tesla V100S PCIe 32 GB Specs.

Inference. For the inference stage, every pair of a context sentence and a candidate with the matching short form in the dictionary constitutes an input to the Next Sentence Prediction task. The language model produces a score for each candidate and we select the one with the highest score as the final predicted output.

4.5.3 Metrics

Acronym disambiguation can be seen as a classification problem, where the input is (1) a dictionary of acronyms and (2) a sentence with an acronym. Each long form for that acronym from the dictionary is considered a class, and the acronym disambiguation has to choose the correct class. We evaluate the models by precision, recall, and macro F1. There are two ways to calculate the macro F1: "F1 of Averages" and "Averaged F1". The first computes the F1 value over the arithmetic means of precision and recall, while the second computes the F1 value for each class, and then averages them. Some prior works adopt the first method. However, this method gives a higher weight to popular classes, and it may thus unfairly yield a high score if the model works well on these popular classes only [195]. Therefore, we use the Averaged F1 across classes as

our metric, which is more robust towards the error type distribution. That is:

$$\text{Precision}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}, i \in \{1, 2, ..., n\} \tag{4.3}$$

$$\text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}, i \in \{1, 2, ..., n\} \tag{4.4}$$

$$\text{F1} = \frac{1}{n} \sum_{i=1}^{n} \frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{4.5}$$

4.5.4 Results

4.5.4.1 Overall Performance

Model	General				Scientific				Biomedical				Avg	
	Dev		Test		Dev		Test		Dev		Test			
	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc
BM25 [228]	29.9	32.6	35.5	25.8	14.1	5.4	17.1	10.7	13.1	8.3	17.0	14.3	21.1	16.2
FastText [20]	11.3	12.9	18.7	12.7	3.3	0.9	5.7	2.5	0.2	0.1	1.3	0.7	6.8	5.0
MadDog [271]	28.1	11.7	29.9	23.1	17.8	15.5	22.4	17.9	33.8	19.3	41.2	35.9	28.9	20.6
BERT [59]	32.3	32.5	37.7	28.2	15.1	5.8	17.6	9.3	3.1	1.3	3.5	2.1	18.2	13.2
Popularity-Ours	35.2	39.1	39.0	43.2	5.5	22.9	4.9	12.3	46.0	**61.3**	49.9	54.0	30.1	38.8
AcroBERT	**63.9**	**68.0**	**69.1**	**67.7**	**28.1**	**34.6**	**32.2**	**28.7**	**53.9**	57.4	**57.1**	**56.4**	**50.7**	**52.1**

Table 4.6: Performances of the unsupervised setting across different models, measured by macro F1 and Accuracy.

	General				Scientific				Biomedical				Avg	
	Dev		Test		Dev		Test		Dev		Test			
	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc
BERT [59]	53.8	70.7	54.9	53.1	13.5	9.9	14.3	10.4	9.8	12.4	9.4	7.5	26.0	27.3
SciBERT [15]	32.4	38.6	33.6	27.7	22.7	19.4	23.4	17.7	31.2	35.6	31.0	28.3	29.5	27.9
BioBERT [152]	26.0	23.6	25.7	20.3	11.2	9.7	12.4	9.0	24.0	21.8	20.2	16.8	19.9	16.9
AcroBERT	**68.6**	**81.3**	**71.7**	**74.1**	**31.7**	**37.8**	**34.2**	**27.7**	**56.9**	**62.7**	**60.5**	**58.9**	**53.9**	**57.0**

Table 4.7: Performances of fine-tuned setting across different models, measured by macro F1 and Accuracy.

Unsupervised Setting. Table 4.6 shows the experimental results in the unsupervised setting. We first observe that our AcroBERT significantly outperforms the baselines across the three domains. For example, AcroBERT can improve the original BERT by 32.5 absolute percentage points of F1 on average on our benchmark. Second, the naive popularity method comes second on this benchmark, most likely because it contains a limited number of overshadowed terms. However, it performs badly on the scientific dataset. We assume that this is because this dataset contains 68.7% of overshadowed terms (as shown in Table 4.5).

Besides, we conduct experiments on existing datasets, namely UAD [47] and SciAD [272]. Our method performs consistently well, as shown in Table 4.8.

Model	UAD		SciAD		Biomedical-UMLS		Avg	
	F1	Acc	F1	Acc	F1	Acc	F1	Acc
BERT [59]	**89.3**	91.1	54.1	57.2	38.0	32.7	60.5	60.3
SciBERT [15]	74.8	78.4	**65.6**	71.7	54.9	50.3	65.1	66.8
BioBERT [152]	66.2	68.2	19.7	22.4	37.4	31.4	41.1	40.7
AcroBERT	88.8	**93.7**	58.0	**72.0**	**67.5**	**65.3**	**71.4**	**77.0**

Table 4.8: Performances on benchmarks with fewer candidates, measured by macro F1 and Accuracy.

Model	Popular	Overshadowed	Avg
BERT [59]	13.3	12.7	13.0
SciBERT [15]	11.6	8.1	9.9
BioBERT [152]	2.1	1.0	1.6
AcroBERT	**61.9**	**33.4**	**47.7**

Table 4.9: Robustness evaluation of overshadowed entities on General test set,measured by Accuracy.

Fine-tuned Setting. In this experiment, every pre-trained language model is fine-tuned on the training set by the triplet loss, as introduced in the pre-training step. Negatives are randomly sampled from ambiguous long forms for the correct label, and the results are shown in Table 4.7. BERT, SciBERT, and BioBERT perform better in their respective fields. However, our AcroBERT achieves the best result across the three fields on average, which demonstrates the effectiveness of the pre-training strategy. One might think that it is unfair that AcroBERT uses the pre-training corpus, while the other models do not. However, there is no other pre-trained model for general disambiguation. Our approach is the first that capitalizes on large-scale corpora and pre-training.

4.5.4.2 Robustness Evaluation

Our GLADIS benchmark is more challenging than existing acronym disambiguation datasets due to the much larger acronym dictionary, which means more candidates per acronym. To measure the robustness of acronym disambiguation systems against more candidates, we sort the samples in the dataset in descending order of the number of candidates per acronym, and divide them evenly into 10 chunks. For example, samples in the first chunk have 1.58 candidates on average while that number is 2159 for the last chunk. The experimental results are shown in Figure 4.4. As expected, the performance of BERT and AcroBERT decreases as the number of candidates increases. However, AcroBERT consistently outperforms BERT on each data chunk, which shows that AcroBERT is able to select the correct long form among the numerous candidates.

Moreover, the challenge with our GLADIS benchmark comes from overshadowed samples, which are harder to disambiguate. To validate the robustness of the models,

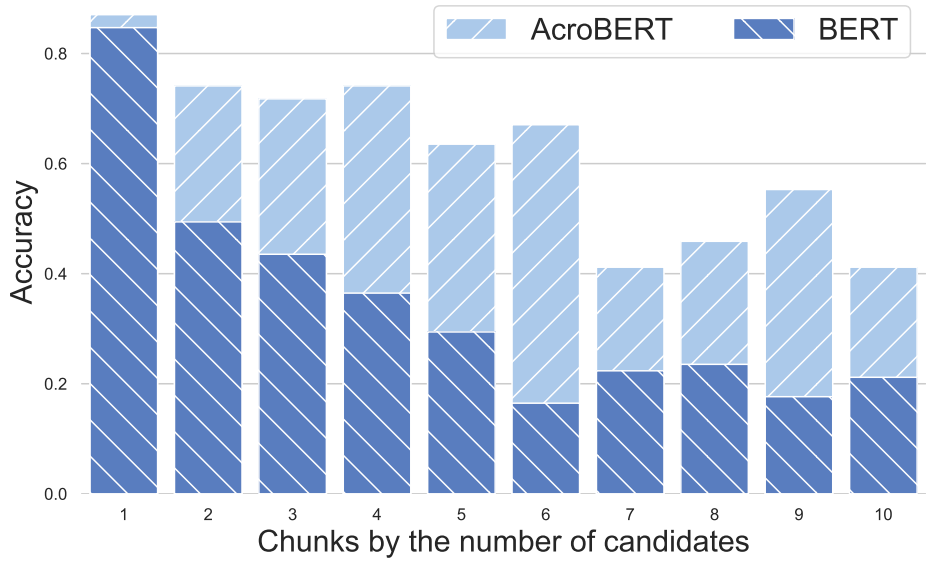

Figure 4.4: Robustness evaluation of hard samples on the General test set. The samples are divided evenly into ten chunks according to the number of candidates of each sample.

we divide the General test set into two parts: Popular and Overshadowed, as described in Section 4.3.2. Next, we compare different language models in the unsupervised setting. As shown in Table 4.9, our AcroBERT performs best on both the Popular and the Overshadowed subset. We conclude that AcroBERT is more robust against ambiguous and overshadowed samples in acronym disambiguation task.

4.5.5 Case Study

Table 4.10 shows case studies of the outputs by BERT and AcroBERT. BERT often uses the memorized correlations of tokens for reasoning and this can cause errors. For example, *External Commercial Borrowings* are loans in India made by non-resident lenders in foreign currency to Indian borrowers [3]. BERT can determine this correct long form probably with help of the key phrase *"external financing"*. For the third case, *Peptic Ulcer Disease* is more consistent with the input context. However, BERT fails on it while AcroBERT benefits from the pre-training strategy and is able to distinguish different candidates based on contexts. For the fourth sample, both methods fail, most likely because of the low frequency of the long forms and the uninformative contexts.

[3] https://en.wikipedia.org/wiki/External_commercial_borrowing

Acronym	Long Form	Context	BERT	AcroBERT
ECB	*European Central Bank*	*Being made to bring the main road network in Romania in the European corridors. There have been initiated several projects to modernize the network of **ECB** corridors, financed from ispa funds and state-guaranteed loans from international financial institutions. Government seeks external financing or public-private partnerships for other road network upgrades , especially*	*External Commercial Borrowing*	*European Central Bank*
PR	*Public Relations*	*A whistleblower like monologist Mike Daisey gets targeted as a scapegoat who must be discredited and diminished in the public 's eye. More often than not, **PR** is a preemptive process. Celebrity publicists are paid lots of money to keep certain stories out of the news.*	*Preemptive-Resume*	*Public Relations*
PUD	*Peptic Ulcer Disease*	*Tumors cause an overactivation of these hormone-producing glands, leading to serious health problems such as severe **PUD** (due to gastrin hypersecretion, which stimulates secretion of hydrochloric acd).*	*Psychogenic Urinary Dysfunction*	*Peptic Ulcer Disease*
WFC	*Walsall F.C.*	*Injury during a game against Norwich city on the 13 march 2010, forcing him to miss Huddersfields next five games. He made his return against **WFC** on the 13 April 2010 , coming on as a 75th minute substitute and scoring a stoppage time winner to make the score 4a3 to town.*	*Wide Free Choice*	*World Fighting Championships*

Table 4.10: Case study of predicted results by BERT and AcroBERT.

4.6 Conclusion

In this paper, we have presented GLADIS, a challenging benchmark for Acronym Disambiguation, which includes a larger dictionary, three datasets from the general, scientific, and biomedical domains, and a large-scale pre-training corpus. We have also proposed AcroBERT, a BERT-based model that is pre-trained on our collected acronym documents, which can significantly outperform other baselines across multiple domains, and which is more robust in the presence of very ambiguous acronyms and overshadowed samples. For future work, we aim to enhance the performance of AcroBERT on the overshadowed cases, which is crucial for the acronym disambiguation task.

5 Robustness: Imputing Out-of-vocabulary Embeddings

State-of-the-art NLP systems represent inputs with word embeddings, but these are brittle when faced with Out-of-Vocabulary (OOV) words. To address this issue, we follow the principle of mimick-like models to generate vectors for unseen words, by learning the behavior of pre-trained embeddings using only the surface form of words. We present a simple contrastive learning framework, LOVE, which extends the word representation of an existing pre-trained language model (such as BERT), and makes it robust to OOV with few additional parameters. Extensive evaluations demonstrate that our lightweight model achieves similar or even better performances than prior competitors, both on original datasets and on corrupted variants. Moreover, it can be used in a plug-and-play fashion with FastText and BERT, where it significantly improves their robustness.

5.1 Introduction

Word embeddings represent words as vectors [177, 179, 204]. They have been instrumental in neural network approaches that brought impressive performance gains to many natural language processing (NLP) tasks. These approaches use a fixed-size vocabulary. Thus they can deal only with words that have been seen during training. While this works well on many benchmark datasets, real-word corpora are typically much noisier and contain Out-of-Vocabulary (OOV) words, i.e., rare words, domain-specific words, slang words, and words with typos, which have not been seen during training. Model performance deteriorates a lot with unseen words, and minor character perturbations can flip the prediction of a model [162, 14, 257, 126]. Simple experiments (Figure 5.1) show that the addition of typos to datasets degrades the performance for text classification models considerably.

To alleviate this problem, pioneering work pre-trained word embeddings with morphological features (sub-word tokens) on large-scale datasets [286, 20, 108, 307]. One of the most prominent works in this direction is FastText [20], which incorporates

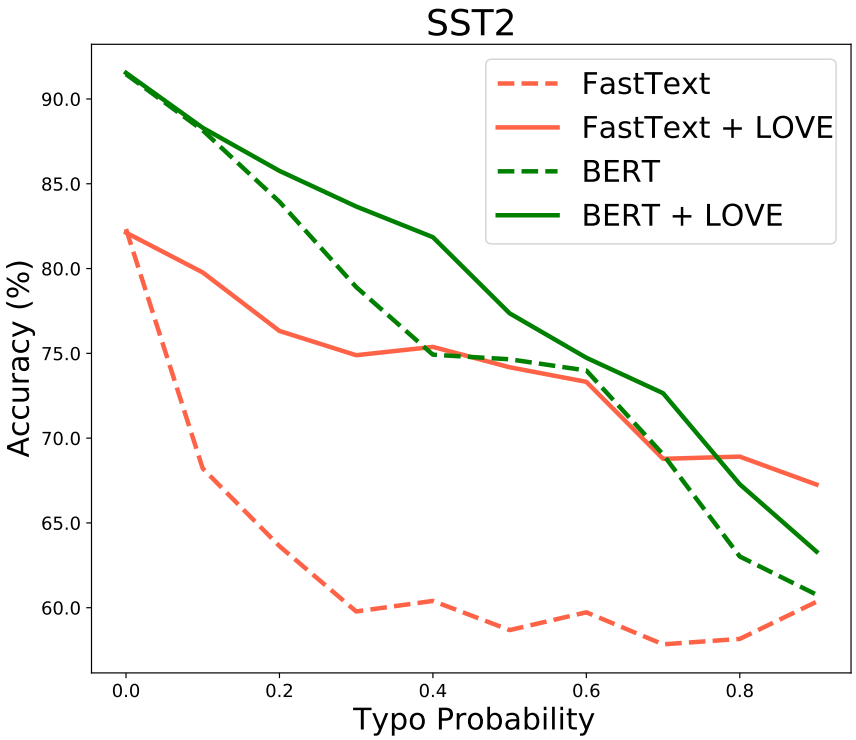

Figure 5.1: Performances of existing word embeddings as we gradually add typos to the datasets. Using our model, LOVE, to produce vectors for OOV words makes the models more robust.

character n-grams into the skip-gram model. With FastText, vectors of unseen words can be imputed by summing up the n-gram vectors. However, these subword-level models come with great costs: the requirements of pre-training from scratch and high memory footprint. Hence, several simpler approaches have been developed, e.g., MIMICK [210], BoS [310] and KVQ-FH [231]. These use only the surface form of words to generate vectors for unseen words, through learning from pre-trained embeddings.

Although MIMICK-like models can efficiently reduce the parameters of pre-trained representations and alleviate the OOV problem, two main challenges remain. First, the models remain bound in the trade-off between complexity and performance: The original MIMICK is lightweight but does not produce high-quality word vectors consistently. BoS and KVQ-FH obtain better word representations but need more parameters. Second, these models cannot be used with existing pre-trained language models such as BERT. It is these models, however, to which we owe so much progress in the domain [205, 60, 300, 169]. To date, these high-performant models are still fragile when dealing with rare words [236], misspellings [257] and domain-specific words [68].

We address these two challenges head-on: we design a new contrastive learning

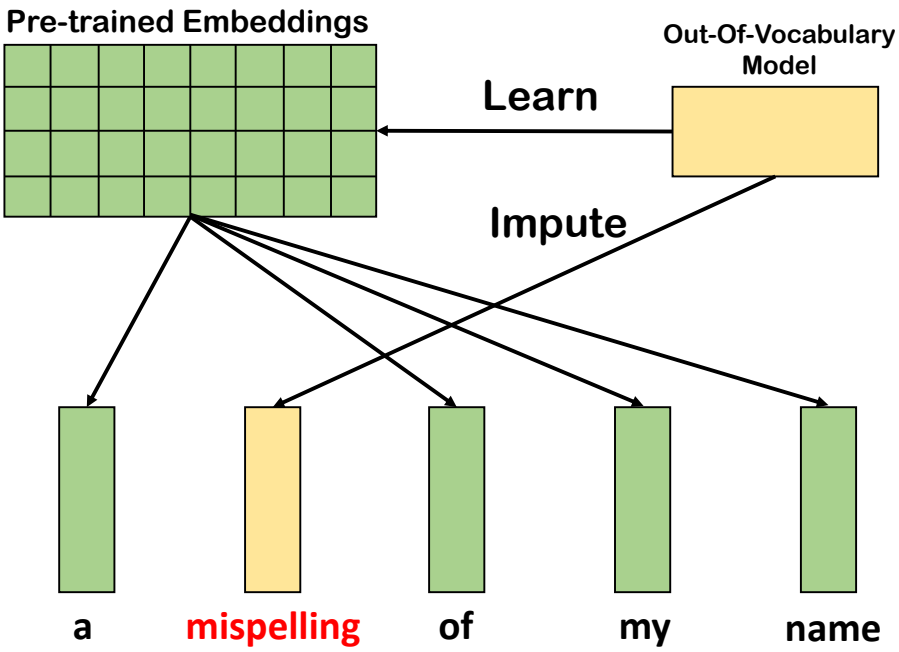

Figure 5.2: Our lightweight OOV model, LOVE, learns the behavior of pre-trained embeddings (e.g., FastText and BERT), and is then able to impute vectors for unseen words. LOVE can enhance the robustness of existing word representations in a plug-and-play fashion.

framework to learn the behavior of pre-trained embeddings, dubbed LOVE, **L**earning **O**ut-of-**V**ocabulary **E**mbeddings. Our model builds upon a memory-saving mixed input of character and subwords instead of n-gram characters. It encodes this input by a lightweight Positional Attention Module. During training, LOVE uses novel types of data augmentation and hard negative generation. The model is then able to produce high-quality word representations that are robust to character perturbations, while consuming only a fraction of the cost of existing models. For instance, LOVE with 6.5M parameters can obtain similar representations as the original FastText model with more than 900M parameters. What is more, our model can be used in a plug-and-play fashion to robustify existing language models. We find that using LOVE to produce vectors for unseen words improves the performance of FastText and BERT by around 1.4-6.8 percentage points on noisy text – without hampering their original capabilities (As shown in Figure 5.2).

In the following, Section 5.2 discusses related work, Section 5.3 introduces preliminaries, Section 5.4 presents our approach, Section 5.5 shows our experiments, and Section 5.6 concludes. The appendix contains additional experiments and analyses. Our code and data is available at GitHub [1].

[1] https://github.com/tigerchen52/LOVE

5.2 Related Work

5.2.1 Character-level Embeddings

To address OOV problems, some approaches inject character-level features into word embeddings during the pre-training [286, 31, 20, 108, 137, 157, 268, 209, 316, 307, 114]. One drawback of these methods is that they need to pre-train on a large-scale corpus from scratch. Therefore, simpler models have been developed, which directly mimic the well-trained word embeddings to impute vectors for OOV words. Some of these methods use only the surface form of words to generate embeddings for unseen words [210, 310, 231, 79, 129], while others use both surface and contextual information to create OOV vectors [234, 235]. In both cases, the models need an excessive number of parameters. FastText, e.g., uses ˜2 million n-gram characters to impute vectors for unseen words.

5.2.2 Pre-trained Language Models

Currently, the state-of-the-art word representations are pre-trained language models, such as ELMo [205], BERT [60] and XLnet [300], which adopt subwords to avoid OOV problems. However, BERT is brittle when faced with rare words [236] and misspellings [257]. To make BERT more robust, CharacterBERT [68] and CharBERT [175] infuse character-level features into BERT and pre-train the variant from scratch. This method can significantly improve the performance and robustness of BERT, but requires pre-training an adapted transformer on a large amount of data. Another work on combating spelling mistakes recommends placing a word corrector before downstream models [215], which is effective and reusable. The main weakness of this method is that an error generated by the word corrector propagates to downstream tasks. For example, converting *"aleph"* to *"alpha"* may break the meaning of a mathematical statement. And indeed, using the word corrector consistently leads to a drop (0.5-2.0 percentage points) in BERT's performance on the SST dataset [249].

5.2.3 Contrastive Learning

The origin of contrastive learning can be traced back to the work [13, 24]. This method has achieved outstanding success in self-supervised representation learning for images [194, 111, 104, 42, 93]. The contrastive learning framework learns representations from unlabeled data by pulling positive pairs together and pushing negative pairs apart. For training, the positive pairs are often obtained by taking two randomly augmented versions of the same sample and treating the other augmented examples within a minibatch as negative examples [43, 42]. The most widely used loss is the infoNCE loss (or contrastive loss) [111, 171, 42, 104]. Although many approaches adopt contrastive learning to represent sentences [88, 295, 82], it has so far not been applied to word representations.

	Input	Encoder	Loss
MIMICK [210]	character sequence {s,p,e,l,l}	RNNs	$\mathscr{L}_{\text{dis}}$
BoS [310]	n-gram subword {spe,pel,ell}	SUM	$\mathscr{L}_{\text{dis}}$
KVQ-FH [231]	adapted n-gram subword {spe,pel,ell}	Attention	$\mathscr{L}_{\text{dis}}$

Table 5.1: Details of different mimick-like models, with the word spell as an example.

5.3 Preliminaries

5.3.1 Mimick-like Model

Given pre-trained word embeddings, and given an OOV word, the core idea of MIMICK [210] is to impute an embedding for the OOV word using the surface form of the word, so as to mimic the behavior of the known embeddings. BoS [310], KVQ-FH [231], Robust Backed-off Estimation [79], and PBoS [129] work similarly, and we refer to them as mimick-like models.

Formally, we have a fixed-size vocabulary set $\mathscr{V}$, with an embedding matrix $\mathbf{W} \in \mathbb{R}^{|\mathscr{V}| \times m}$, in which each row is a word vector $\mathbf{u}_w \in \mathbb{R}^m$ for the word w. A mimick-like model aims to impute a vector $\mathbf{v}_w$ for an arbitrary word $w \notin \mathscr{V}$. The training objective of mimick-like models is to minimize the expected distance between $\mathbf{u}_w$ and $\mathbf{v}_w$ pairs:

$$\mathscr{L}_{\text{dis}} = \frac{1}{|\mathscr{V}|} \sum_{w \in \mathscr{V}} \psi(\mathbf{u}_w, \mathbf{v}_w) \tag{5.1}$$

Here, $\psi(\cdot)$ is a distance function, e.g., the Euclidean distance $\psi = \|\mathbf{u}_w - \mathbf{v}_w\|_2^2$ or the cosine distance $\psi = 1 - \cos(\mathbf{u}_w, \mathbf{v}_w)$. The vector $\mathbf{v}_w$ is generated by the following equation:

$$\mathbf{v}_w = \phi(\zeta(w)), \text{ for } w \in \mathscr{V} \text{ or } w \notin \mathscr{V} \tag{5.2}$$

Here, $\zeta(\cdot)$ is a function that maps w to a list of subunits based on the surface form of the word (e.g., a character or subword sequence). After that, the sequence is fed into the function $\phi(\cdot)$ to produce vectors, and the inside structure can be CNNs, RNNs, or a simple summation function. After training, the model can impute vectors for arbitrary words. Table 5.1 shows some features of three mimick-like models.

5.3.2 Contrastive Learning

Contrastive learning methods have achieved significant success for image representation [194, 42]. The core idea of these methods is to encourage learned representations

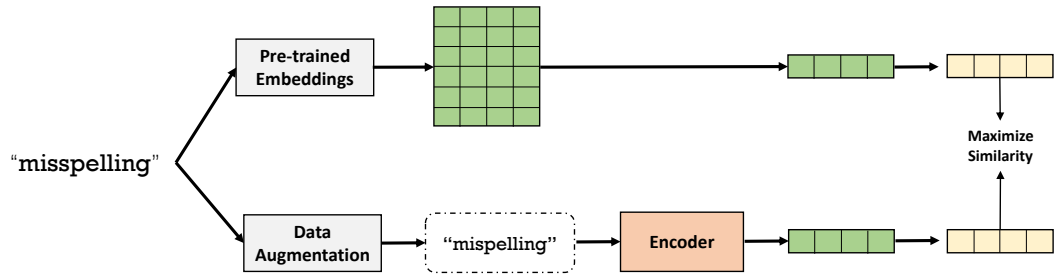

Figure 5.3: The framework of LOVE with an example of the word `misspelling`.

for positive pairs to be close, while pushing representations from sampled negative pairs apart. The widely used contrastive loss [111, 171, 42, 104] is defined as:

$$\ell_{cl} = -\log \frac{e^{\text{sim}(\mathbf{u}_i^\top \mathbf{u}^+)/\tau}}{e^{\text{sim}(\mathbf{u}_i^\top \mathbf{u}^+)/\tau} + \sum e^{\text{sim}(\mathbf{u}_i^\top \mathbf{u}^-)/\tau}} \tag{5.3}$$

Here, τ is a temperature parameter, $\text{sim}(\cdot)$ is a similarity function such as cosine similarity, and (u_i, u^+), (u_i, u^-) are positive pairs and negative pairs, respectively (assuming that all vectors are normalized). During training, positive pairs are usually obtained by augmentation for the same sample, and negative examples are the other samples in the mini-batch. This process learns representations that are invariant against noisy factors to some extent.

5.4 Our Approach: LOVE

LOVE (Learning Out-of-Vocabulary Embeddings) draws on the principles of contrastive learning to maximize the similarity between target and generated vectors, and to push apart negative pairs. An overview of our framework is shown in Figure 5.3. It is inspired by work in visual representation learning [42], but differs in that one of the positive pairs is obtained from pre-trained embeddings instead of using two augmented versions. We adopt five novel types of word-level augmentations and a lightweight Positional Attention Module in this framework. Moreover, we find that adding hard negatives during training can effectively yield better representations. We removed the nonlinear projection head after the encoder layer, because its improvements are specific to the representation quality in the visual field. Furthermore, our approach is not an unsupervised contrastive learning framework, but a supervised learning approach.

Our framework takes a word from the original vocabulary and uses data augmentation to produce a corruption of it. For example, `"misspelling"` becomes `"mispelling"` after dropping one letter `"s"`. Next, we obtain a target vector from the pre-trained embeddings for the original word, and we generate a vector for the

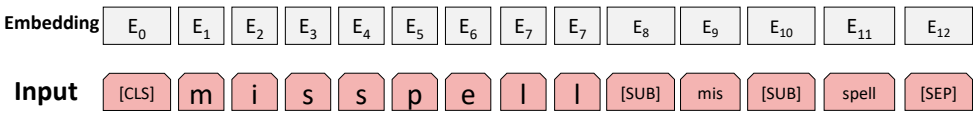

Figure 5.4: An illustration of our Mixed input for the word `misspell`.

corrupted word. These two vectors are a pair of positive samples, and we maximize the similarity between them while making the distance of negative pairs (other samples in the same mini-batch) as large as possible. As mentioned before, we use the contrastive loss as an objective function (Eq 5.3). There are five key ingredients in the framework that we will detail in the following (similar to the ones in Table 5.1): the Input Method, the Encoder, the Loss Function, our Data Augmentation, and the choice of Hard Negatives.

5.4.1 Input Method

Our goal is to use the surface form to impute vectors for words. The question is thus how to design the function $\zeta(\cdot)$ mentioned in Section 5.3.1 to represent each input word. MIMICK [210] straightforwardly uses the character sequence (see Table 5.1). This, however, loses the information of morphemes, i.e., sequences of characters that together contribute a meaning. Hence, FastText [20] adopts character n-grams. Such n-grams, however, are highly redundant. For example, if we use substrings of length 3 to 5 to represent the word `misspelling`, we obtain a list with 24 n-gram characters – while only the substrings {`mis, spell, ing`} are the three crucial units to understand the word. Hence, like BERT, we use WordPiece [293] with a vocabulary size of around 30000 to obtain meaningful subwords of the input word. For the word `misspelling`, this yields {`miss`, `##pel`, `##ling` }. However, if we just swap two letters (as by a typo), then the sequence becomes completely different: {`mi`, `##sp`, `##sell`, `##ing` }. Therefore, we use both the character sequence and subwords (Figure 5.4).

We shrink our vocabulary by stemming all words and keeping only the base form of each word, and by removing words with numerals. This decreases the size of vocabulary from 30 000 to 21 257 without degrading performance too much.

5.4.2 Encoder

Let us now design the function $\phi(\cdot)$ mentioned in Section 5.3.1. We are looking for a function that can encode both local features and global features. Local features are character n-grams, which provide robustness against minor variations such as character swaps or omissions. Global features combine local features regardless of their distance. For the word `misspelling`, a pattern of prefix and suffix `mis+ing` can be obtained by combining the local information at the beginning and the end of the word. Conventional CNNs, RNNs, and self-attention cannot extract such local

and global information at the same time. Therefore, we design a new **Positional Attention Module**. Suppose we have an aforementioned mixed input sequence and a corresponding embedding matrix $\mathbf{V} \in \mathbb{R}^{|\mathcal{V}| \times d}$ where d is the dimension of vectors. Then the input can be represented by a list of vectors: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^{n \times d}$ where n is the length of the input. To extract local information, we first adopt positional attention to obtain n-gram features, and then feed them into a conventional self-attention layer to combine them in a global way. This can be written as:

$$\bar{\mathbf{X}} = \mathrm{SA}(\mathrm{PA}(\mathbf{X}))\,\mathbf{W}^O \tag{5.4}$$

Here, SA is a standard multi-head self-attention and PA is a positional attention. $\mathbf{W}^O \in \mathbb{R}^{d_V \times d_O}$ is a trainable parameter matrix, where d_V are the dimensions of values in SA and PA, and d_O is that of $\bar{\mathbf{X}}$. As for the Positional Attention, we adopt absolute sinusoidal embeddings [270] to compute positional correlations:

$$\mathrm{PA}(\mathbf{X}) = \mathrm{Softmax}\left(\frac{\mathbf{P}\mathbf{P}^\top}{\sqrt{d}}\right)(\mathbf{X}\mathbf{W}^V) \tag{5.5}$$

Here, $\mathbf{P} \in \mathbb{R}^{n \times d}$ are the position embeddings, and $\mathbf{W}^V \in \mathbb{R}^{d \times d_V}$ are the corresponding parameters.

5.4.3 Loss Function

In this section, we focus on the loss function $\mathscr{L}(\cdot)$. Mimick-like models often adopt the mean squared error (MSE), which tries to give words with the same surface forms similar embeddings. However, the MSE only pulls positive word pairs closer, and does not push negative word pairs apart. Therefore, we use the contrastive loss instead (Equation 5.3). **(author?)** [282] found that the contrastive loss optimizes two key properties: *Alignment* and *Uniformity*. The Alignment describes the expected distance (closeness) between positive pairs:

$$\ell_{\text{align}} \triangleq \mathop{\mathbb{E}}_{(x,y)\sim p_{\text{pos}}} \psi(\mathbf{u}_x, \mathbf{u}_y) \tag{5.6}$$

Here, p_{pos} is the distribution of positive pairs. The Uniformity measures whether the learned representations are uniformly distributed in the hypersphere:

$$\ell_{\text{uniform}} \triangleq \log \mathop{\mathbb{E}}_{(x,y)\overset{i.i.d.}{\sim} p_{\text{data}}} e^{-t\cdot\psi(\mathbf{u}_x,\mathbf{u}_y)} \tag{5.7}$$

Here, p_{data} is the data distribution and $t > 0$ is a parameter. The two properties are consistent with our expected word representations: positive word pairs should be kept close and negative word pairs should be far from each other, finally scattered over the hypersphere.

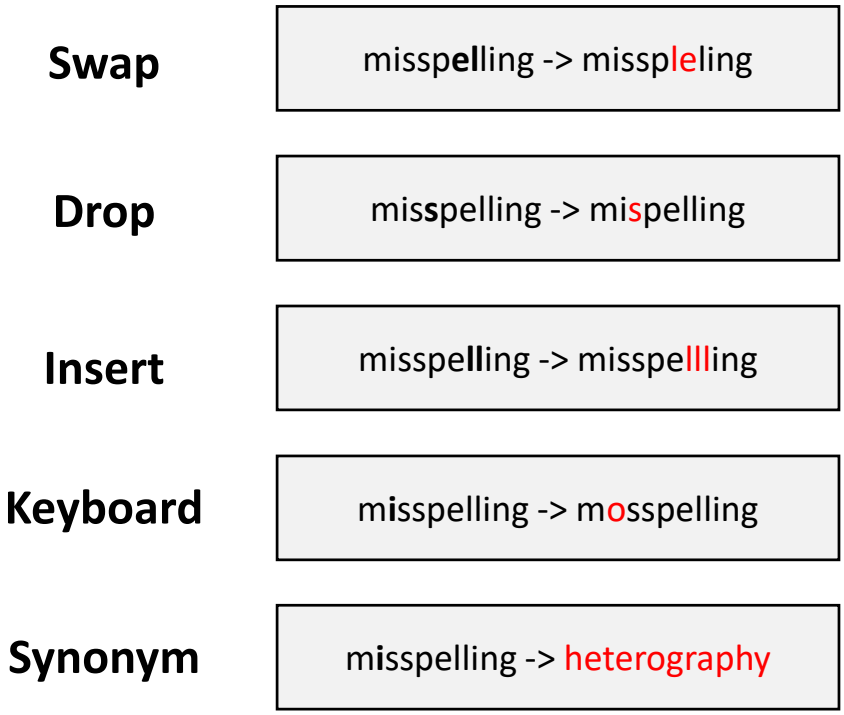

Figure 5.5: Illustrations of different augmentations for the word `misspelling`.

5.4.4 Data Augmentation and Hard Negatives

Our positive word pairs are generated by data augmentation, which can increase the amount of training samples by using existing data. We use various strategies (Figure 5.5) to increase the diversity of our training samples: (1) Swap two adjacent characters, (2) Drop a character, (3) Insert a new character, (4) Replace a character according to keyboard distance, (5) Replace the original word by a synonymous word. The first four augmentations are originally designed to protect against adversarial attacks [215]. We add the synonym replacement strategy to keep semantically similar words close in the embedding space – something that cannot be achieved by the surface form alone. Specifically, a set of synonyms is obtained by retrieving the nearest neighbors from pre-trained embeddings like FastText.

Negative word pairs are usually chosen randomly from the mini-batch. However, we train our model to be specifically resilient to *hard negatives (or difficult negatives)*, i.e., words with similar surface forms but different meanings (e.g., *misspelling* and *dispelling*). To this end, we add a certain number of hard negative samples (currently 3 of them) to the mini-batch, by selecting word pairs that are not synonyms and have a small edit distance.

5.4.5 Mimicking Dynamical Embeddings

Pre-trained Language Models (e.g., ELMo [205] and BERT [60]) dynamically generate word representations based on specific contexts, which cannot be mimicked directly. To this end, we have two options: We can either learn the behavior of the *input embeddings* in BERT before the multi-layer attentions or mimic the static *distilled embeddings* [22, 99].

We use the BERT as an example to explain these two methods. Suppose we have a subword sequence after applying WordPiece to a sentence: $W = \{w_1, w_2, ..., w_n\}$. For the subword sequence W, BERT first represents it as a list of subword embeddings: $\mathbf{E}^{in} = \{\mathbf{e}_1^{sub}, \mathbf{e}_2^{sub}, ..., \mathbf{e}_n^{sub}\}$. We refer to this static representation as the Input Embedding of BERT, and we can use our model to mimic the behavior of this part. We call this method *mimicking input embeddings*. For ease of implementation, we learn only from the words that are not separated into pieces. After that step, BERT applies a multi-layer multi-head attention to the input embeddings E^{in}, which yields a contextual representation for each subword: $\mathbf{E}^{out} = \{\mathbf{e}_1^{out}, \mathbf{e}_2^{out}, ..., \mathbf{e}_n^{out}\}$. However, these contextual representations vary according to the input sentence and we cannot learn from them directly. Instead, we choose to mimic the distilled static embeddings from BERT, which are obtained by pooling (max or average) the contextual embeddings of the word in different sentences. We call this method *mimicking distilled embeddings*. The latter allows for better word representations, while the former does not require training on a large-scale corpus. Our empirical studies show that mimicking distilled embeddings performs only marginally better. Therefore, we decided to rather learn the input embeddings of BERT, which is simple yet effective.

5.4.6 Plug and Play

One of the key advantages of our model is that it can be used as a plug-in for other models. For models with static word embeddings like FastText, one can simply use our model to generate vectors for unseen words. For models with dynamic word embeddings like BERT, if a single word is tokenized into several parts, e.g. `misspelling` = {`miss`, `##pel`, `##ling` }, we regard it as an OOV word. Then, we replace the embeddings of the subwords by a single embedding produced by our model before the attention layer. Our final enhanced BERT model has 768 dimensions and 16M parameters. Note that the BERT-base model has ˜110M parameters and its distilled one has ˜550M parameters.

5.5 Experiments

5.5.1 Evaluation Datasets

There are two main methods to evaluate word representations: Intrinsic and Extrinsic. Intrinsic evaluations measure syntactic or semantic relationships between words directly, e.g., word similarity in word clusters. Extrinsic evaluations measure the

performance of word embeddings as input features to a downstream task, e.g., named entity recognition (NER) and text classification. Several studies have shown that there is no consistent correlation between intrinsic and extrinsic evaluation results [46, 74, 278]. Hence, we evaluate our representation by both intrinsic and extrinsic metrics. Specifically, we use 8 intrinsic datasets (6 word similarity and 2 word cluster tasks): RareWord [173], SimLex [110], MTurk [101], MEN [27], WordSim [3], Simverb [3], AP [7] and BLESS [12]. We use four extrinsic datasets (2 text classification and 2 NER tasks): SST2 [249], MR [199], CoNLL-03 [230] and BC2GM [248]. It is worth noting that the RareWord dataset contains many long-tail words and the BC2GM is a domain-specific NER dataset. All data augmentations and typo simulations are implemented by NLPAUG[2].

5.5.2 Experimental Settings

5.5.2.1 Training of Mimick-like Models

Our target pre-trained embeddings are those from FastText, because they provide a strong baseline. They sum up subword-level information to produce vectors for arbitrary words. We also compare to MIMICK, BoS, and KVQ-FH, which do not train on contextual words. We do not compare to Robust Backed-off Estimation [79] and PBoS [129], because they need larger and more complex models. Robust Backed-off Estimation uses string matching to find the top-k similar words from the entire vocabulary when imputing. Using the same target vectors, the number of parameter of BoS and PBoS are 163M and 316M, respectively. We re-train MIMICK, BoS, and KVQ-FH as baselines according to the published settings. In order to compare at the same parameter level, we use subwords for MIMICK instead of pure characters and adjust the hashing size $H = 40K$ for KVQ-FH.

5.5.2.2 Robustness Evaluations

As for our model, we first lower-case and tokenize each word by using WordPiece [293] with a vocabulary of $30K$ subwords and preprocess them by stemming and removing subwords with numerals. This yields a vocabulary of 21257 words. Each subword is represented by corresponding vectors from our model and we adopt a modified attention model to encode the subword sequence. Specifically, the layer number of this encoder is just 1 for efficiency and the hidden dimension is 300. In each block, the number of attention heads is 1 and we use fixed sinusoidal position embeddings [270] for positional information. To train the contrastive learning framework, we use the open-source tool [174] to augment a word, and use the probabilities $\{0.07, 0.07, 0.07, 0.07, 0.36, 0.36\}$ for six augmentations: swap, drop, insert, keyboard, synonym, no-operation. Hard negatives are generated by edit distance. For each target word, we store the top-100 similar words and insert 3 of them into a mini-batch as hard negatives. The loss function is a standard contrastive loss with temperature $\tau = 0.07$.

[2] https://github.com/makcedward/nlpaug

Hyperparam	SST2	MR	CONLL-03	BC2GM
model	CNN	CNN	BiLSTM+CRF	BiLSTM+CRF
layer	1	1	1	1
kernel	[3,4,5]	[3,4,5]	-	-
filter	100	100	-	-
hidden size	300	300	300	300
optimizer	Adam	Adam	SGD	SGD
dropout	0.5	0.5	0.5	0.5
batch size	50	50	768	768
epoch	5	5	100	100

Table 5.2: Hyperparameters for extrinsic datasets.

The optimizer is Adam and the learning rate is 0.002. The dropout rate is 0.2 and we train the model for 20 epochs in total.

5.5.2.3 Intrinsic and Extrinsic Evaluations

We choose the setting discussed in Section 5.4 to train our model for 20 epochs, and evaluate each intrinsic task based on the vectors that the models produce. As for the extrinsic tasks, we feed word vectors into each neural network and fix them during training. We use CNNs for text classification [306] and BiLSTM+CRF for NER [116]. We compare different embeddings on both intrinsic and extrinsic datasets by using generated vectors. For the word cluster tasks, the produced vectors are clustered by K-Means and then measured by Purity. The hyper-parameters of the extrinsic tasks are shown in Table 5.2. For each dataset, our model is trained with five learning rates $\{5e-3, 3e-3, 1e-3, 8e-4, 5e-4\}$. We select the best one on the development set to report its score on the test set.

To generate a corrupted dataset, we simulate post-OCR errors. We adopt the augmentation tool [174] to corrupt 70% of the original words. To check the robustness of BERT, we directly finetune a BERT-base model using Huggingface [288]. During finetuning, the batch size is 16 and we train 5 epochs. We select the best model among five learning rates $\{9e-5, 7e-5, 5e-5, 3e-5, 1e-5\}$ on the development set and report the score of the model on the test set.

5.5.3 Results on Intrinsic Tasks

Table 5.3 shows the experimental results on 8 intrinsic tasks. Compared to other mimick-like models, our model achieves the best average score across 8 datasets while using the least number of parameters. Specifically, our model performs best on 5 word similarity tasks, and second-best on the word cluster tasks. Although there is a gap between our model and the original FastText, we find our performance acceptable, given that our model is 100x times smaller.

	parameters		Word Similarity						Word Cluster		Avg
	embedding	others	RareWord	SimLex	MTurk	MEN	WordSim	SimVerb	AP	BLESS	
FastText [20]	969M	-	48.1	30.4	66.9	78.1	68.2	25.7	58.0	71.5	55.9
MIMICK [210]	9M	517K	27.1	15.9	32.5	36.5	15.0	7.5	**59.3**	**72.0**	33.2
BoS [310]	500M	-	**44.2**	27.4	55.8	65.5	53.8	22.1	41.8	39.0	43.7
KVQ-FH [231]	12M	-	42.4	20.4	55.2	63.4	53.1	16.4	39.1	42.5	41.6
LOVE	6.3M	200K	42.2	**35.0**	**62.0**	**68.8**	**55.1**	**29.4**	53.2	51.5	**49.7**

Table 5.3: Performance on the intrinsic tasks, measured as Spearman's ρ and purity for word similarity and clustering. Best performance among the mimick-like models in bold, second-best underlined.

	parameters		**SST2**		**MR**		**CoNLL-03**		**BC2GM**		Avg
	embedding	others	original	+typo	original	+typo	original	+typo	original	+typo	
FastText [20]	969M	-	82.3	60.5	73.3	62.2	86.4	66.3	71.8	53.4	69.5
Edit Distance	969M	-	-	67.4	-	68.3	-	76.2	-	66.6	-
MIMICK [310]	9M	517K	69.7	62.3	73.6	61.4	68.0	65.2	56.6	56.7	64.2
BoS [310]	500M	-	79.7	72.6	73.6	**69.5**	**79.5**	68.6	**66.4**	61.5	71.5
KVQ-FH [231]	12M	-	77.8	71.4	72.9	66.5	73.1	**70.4**	46.2	53.5	66.5
LOVE	6.3M	200K	**81.4**	**73.2**	**74.4**	66.7	78.6	69.7	64.7	**63.8**	**71.6**

Table 5.4: Performance on the extrinsic tasks, measured as accuracy and F1 (five runs of different learning rates) for text classification and NER, respectively. Typos are generated by simulated errors of an OCR engine [174]. The speed of producing word vectors with Edit Distance and LOVE is *380s/10K* words and *0.9s/10K* words, respectively.

5.5.4 Results on Extrinsic Tasks

Table 5.4 shows the results on four downstream datasets and their corrupted version. In this experiment, we introduce another non-trivial baseline: Edit Distance. For each corrupted word, we find the most similar word from a vocabulary using edit distance and then use the pre-trained vectors of the retrieved word. Considering the time cost, we only use the first 20K words appearing in FastText (2M words) as reference vocabulary.

The typo words are generated by simulating post-OCR errors. For the original datasets, our model obtains the best results across 2 datasets and the second-best on NER datasets compared to other mimick-like models. For the corrupted datasets, the performance of the FastText model decreases a lot and our model is the second best but has very close scores with BoS consistently. Compared to other mimick-like models, our model with 6.5M achieves the best average score. Although Edit Distance can effectively restore the original meaning of word, it is 400x times more time-consuming than our model.

5.5.5 Robustness Evaluation

In this experiment, we evaluate the robustness of our model by gradually adding simulated post-OCR typos [174]. Table 5.5 shows the performances on SST2 and CoNLL-03 datasets. We observe that our model can improve the robustness of the original embeddings without degrading their performance. Moreover, we find our model can make FastText more robust compared to other commonly used methods against

Typo Probability	SST2 original	10%	30%	50%	70%	90%	CoNLL-03 original	10%	30%	50%	70%	90%	Avg
Static Embeddings													
FastText	**82.3**	68.2	59.8	56.7	57.8	60.3	**86.4**	81.6	78.9	73.9	70.2	63.4	70.0
FastText + LOVE	82.1	**79.8**	**74.9**	**74.2**	**68.8**	**67.2**	86.3	**84.7**	**81.8**	**77.5**	**73.1**	**71.3**	**76.8**
Dynamical Embeddings													
BERT	**91.5**	88.2	78.9	74.7	69.0	60.1	**91.2**	**89.8**	**86.2**	83.4	79.9	76.5	80.7
BERT + LOVE	**91.5**	**88.3**	**83.7**	**77.4**	**72.7**	**63.3**	89.9	88.3	86.1	**84.3**	**80.8**	**78.3**	**82.1**

Table 5.5: Robust evaluation (five runs of different learning rates) on text classification and NER under simulated post-OCR typos. We use uncased and cased BERT-base model for SST2 and CoNLL-03, respectively.

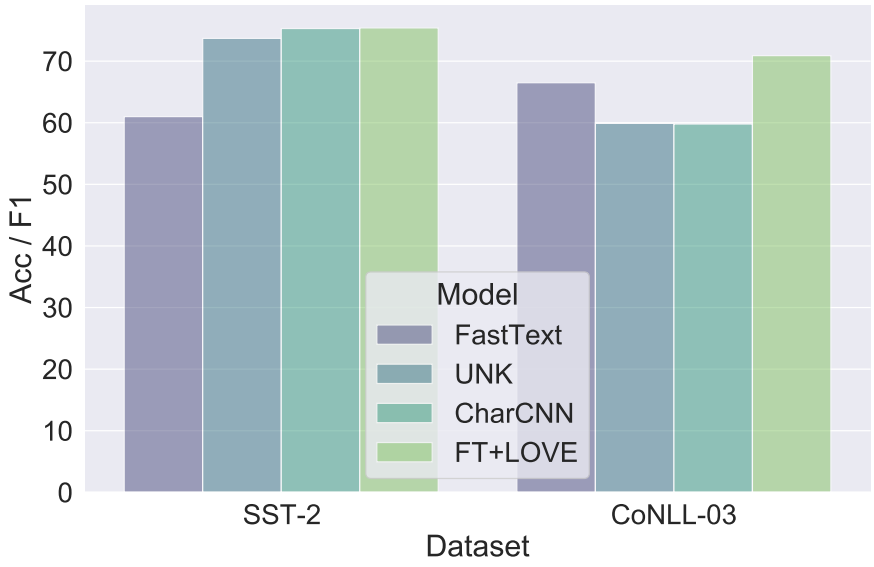

Figure 5.6: Evaluation of different methods based on FastText under typos.

unseen words: a generic `UNK` token or character-level representation of neural networks. Figure 5.6 shows the robustness check of different strategies. FastText+LOVE has a consistent improvement on both SST2 and CoNLL-03 datasets. At the same time, LOVE degrades the performance on the original datasets only marginally if at all.

5.5.6 Qualitative Analysis

To better understand the clusterings produced by LOVE, we chose 15 words from the AP dataset [7], covering three topics (Chemical Substance, Illness, and Occupation). We added 3 corrupted words, `oxgen`, `archiitect` and `leukamia`. Figure 5.7 shows how LOVE, BoS, and KVQ-FH cluster these words (using a PCA projection and K-means). All approaches space out the clusters to some degree. In particular, BoS and KVQ-FH have trouble separating professions and chemical substances. For the corrupted words, only LOVE is able to embed them close enough to their original form, so that they appear in the correct cluster.

	parameters		RareWord	SST2
	embedding	others		
The original LOVE	6.3M	200K	42.2	81.4
Varying the input method				
only use Char	299K	200K	17.7	71.5
only use Subword	6.0M	200K	25.3	76.0
Varying the encoder				
replace PAM with CNN	6.3M	270K	28.4	61.1
replace PAM with RNN	6.3M	517K	27.2	67.2
replace PAM with SA	6.3M	-	36.9	78.7
Varying the loss function				
use MSE	6.3M	200K	34.5	76.0
use $\ell_{au}(\lambda = 1.0)$	6.3M	200K	40.8	80.8
Ablation of data augmentation and hard negatives				
w/out hard negatives	6.3M	200K	37.7	78.6
w/out hard negatives and augmentation	6.3M	200K	37.8	78.2

Table 5.6: Ablation studies for the architecture of LOVE, measured as Spearman's ρ and accuracy, respectively.

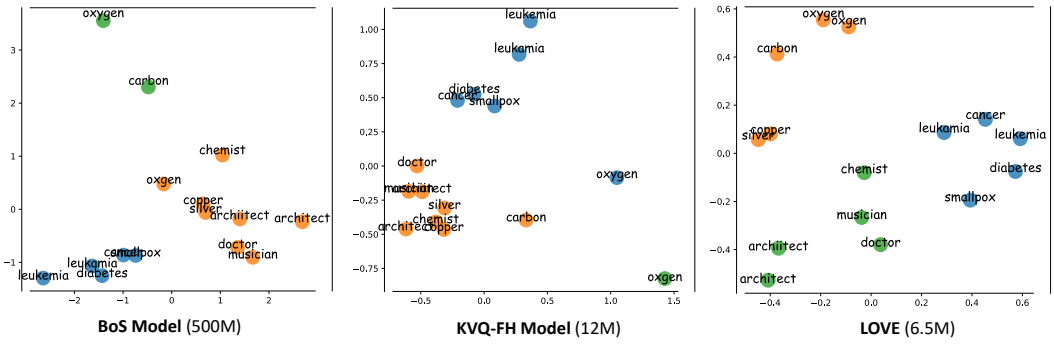

Figure 5.7: PCA visualizations of word vectors generated by LOVE, BoS, and KVQ-FH. Different colors mean different clusters, as predicted by K-means. There are three OOV words: `oxgen`, `architect` and `leukamia`.

5.5.7 Ablation Study

We now vary the components in our architecture (input method, encoder and loss function) to demonstrate the effectiveness of our architecture.

Input Method. To validate the effect of our Mixed Input strategy, we compare it with two other methods: using only the character sequence or only the subword sequence. Table 5.6 shows that the Mixed method achieves better representations, and any removal of char or subword information can decrease the performance.

Encoder. To encode the input sequence, we developed the Positional Attention Module (PAM), which first extracts ngram-like local features and then uses self-attention combine them without distance restrictions. Table 5.6 shows that PAM performs the best, which validates our strategy of incorporating both local and global parts inside a word. At the same time, the number of parameters of PAM is acceptable in comparison. We visualize the attention weights of PAM in Appendix 5.5.10, to show how the encoder extracts local and global morphological features of a word.

Loss Function. LOVE uses the contrastive loss, which increases alignment and uniformity. An existing work proves that optimizing directly these two metrics leads to comparable or better performance than the original contrastive loss [282] . Such a loss function can be written as:

$$\ell_{\text{au}} = \ell_{\text{align}} + \lambda \cdot \ell_{\text{uniform}} \tag{5.8}$$

Here, λ is a hyperparameter that controls the impact of ℓ_{uniform}. We set this value to 1.0 because it achieves the best average score on RareWord and SST2. An alternative is to use the Mean Squared Error (MSE), as in mimick-like models. Table 5.6 compares the performances of these different loss functions. The contrastive loss significantly outperforms the MSE, and there is no obvious improvement by directly using alignment and uniformity.

Data Augmentation and Hard Negatives. In Table 5.6, we observe that the removal of our hard negatives decreases the performance, which demonstrates the importance of semantically different words with similar surface forms. LOVE uses five types of word augmentation.

We find that removing this augmentation does not deteriorate performance too much on the word similarity task, while it causes a 0.4 point drop in the text classification task (the last row in Table 5.6), where data augmentations prove helpful in dealing with misspellings. We further analyze the performance of single and composite augmentations on RareWord and SST2 in the appendix in Figure 5.8. We find that a combination of all five types yields the best results.

5.5.8 Shrinking Our Model

We consider the following four methods to reduce the total parameters of our model:
(1) Matrix Decomposition. The original matrix can be decomposed into two smaller matrices $\mathbf{V} = \mathbf{U} \times \mathbf{M}, \mathbf{U} \in \mathbb{R}^{|\mathcal{V}| \times h}, \mathbf{M} \in \mathbb{R}^{h \times m}$ and $h < m$. Here, we set $m = 300$ and $h = 200$ respectively.

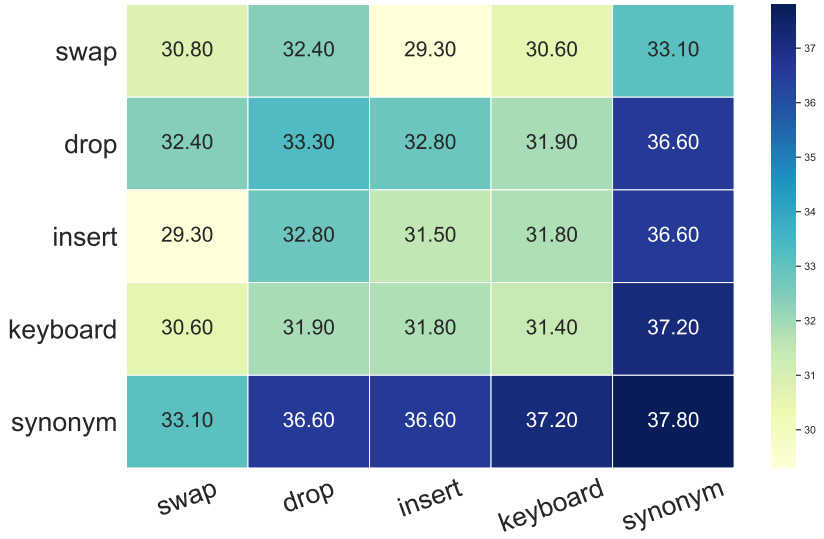

Figure 5.8: Performances of different augmentations on RareWord, measured as Spearman's ρ. Diagonal entries correspond to individual augmentation and off-diagonal entries correspond to composite augmentation.

	SST2			
typos per sentence	typo-0	typo-1	typo-2	typo-3
BERT	**91.5**	77.2	73.2	69.4
Mimicking Input Embeddings				
BERT + Add	91.3	77.2	73.5	70.7
BERT + Linear [79]	91.4	79.6	77.2	72.8
BERT + Replacement	**91.5**	81.4	78.7	73.6
Mimicking Distilled Embeddings				
BERT + Add	91.3	78.8	75.6	72.3
BERT + Linear [79]	91.3	81.4	78.7	73.6
BERT + Replacement	91.4	**81.5**	**78.9**	**73.8**

Table 5.7: Performances of different strategies that work with BERT together, measured as the accuracy among five different learning rates.

(2) Top Subword. We use only the top-k frequent subwords, using the word frequencies from a corpus. We set the parameter $k = 20000$.
(3) Hashing. We use a hashing strategy to share memory for subwords aiming to reduce the parameters. We use a bucket size of 20000.
(4) Preprocessing. The original vocabulary contains plurals and conjugations, therefore we stem all complete words and remove words with numerals, obtaining a new

vocabulary of 21257 words.

Table 5.8 shows that the preprocessing method can reduce parameters very effectively while obtaining a very competitive performance.

	parameters		RareWord	SST2
	embedding	non-embedding		
Original	9M	200K	43.5	80.7
Decomposition	5.6M	200K	38.1	80.3
Top-K	6M	200K	39.2	80.1
Hashing	6M	200K	40.5	80.4
Preprocessing	6.3M	200K	**42.4**	**80.7**

Table 5.8: Performance of different shrinkage strategies, measured as Spearman's ρ and accuracy, respectively. The target vectors are from `fasttext-crawl-300d-2M`.

5.5.9 The performance of mimicking BERT

As described in Section 5.4.5, we can mimic the input or distilled embeddings of BERT. After learning from BERT, we use the vectors generated by LOVE to replace the embeddings of OOV subwords. Finally, these new representations are fed into the multi-layer attentions. We call this method the *Replacement* strategy. To valid its effectiveness, we compare it with two other baselines: **(1) Linear Combination** [79]. For each subword $\mathbf{e}^{sub}$, the generated vectors of word $\mathbf{e}^{word}$ containing the subwords are added to the subword vectors of BERT:

$$\begin{aligned} \mathbf{e}^{new} &= (1-\alpha)\,\mathbf{e}^{sub} + \alpha\,\mathbf{e}^{word} \\ \alpha &= \text{sigmoid}\,(\mathbf{W}\cdot\mathbf{e}^{sub}) \end{aligned} \tag{5.9}$$

where $\mathbf{e}^{sub} \in \mathbb{R}^d$ is a subword vector of BERT, and $\mathbf{e}^{word} \in \mathbb{R}^d$ is a generated vector of our model. $\mathbf{W} \in \mathbb{R}^d$ are trainable parameters.
(2) Add. A generated word vector is directly added to a corresponding subword vector of BERT:

$$\mathbf{e}^{new} = \mathbf{e}^{sub} + \mathbf{e}^{word} \tag{5.10}$$

Table 5.7 shows the result of these strategies. All of them can bring a certain degree of robustness to BERT without decreasing the original capability, which demonstrates the effectiveness of our framework. Second, the replacement strategy consistently performs best. We conjecture that BERT cannot restore a reasonable meaning for those rare and misspelling words that are tokenized into subwords, and our generated vectors can be located nearby the original word in the space. Third, we find mimicking distilled embeddings performs the best while mimicking input embeddings comes close. Considering that the first method needs training on large-scale data, mimicking the input embeddings is our method of choice.

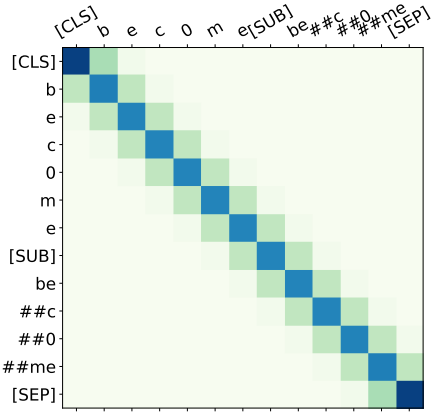

Figure 5.9: Visualization of positional weights for the post-OCR word `bec0me` (the correct one is `become`).

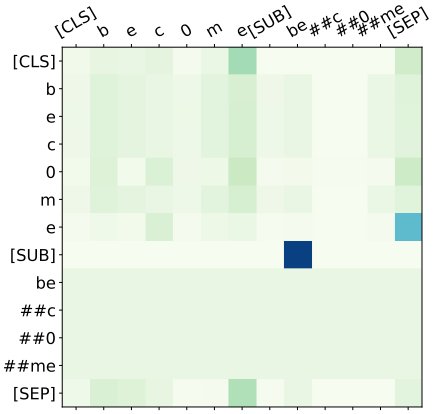

Figure 5.10: Visualization of self-attention weights for the post-OCR word `bec0me`.

5.5.10 Visualization of Encoder

As mentioned before, we combine two types of attention heads (self-attention and positional attention) to encode a subword sequence. Here, we visualize the attention weights on each side and show how they work. Figure 5.9 shows the position-dependent weights. We use sinusoidal functions to generate positional embeddings, and the weights are the dot product between these embeddings. We observe the positional weights tend to the left and right subwords in addition to themselves, which yields trigram representations.

Figure 5.10 shows the self-attention weights which are computed from the trigram subwords of positional attention. Hence, each subword in this figure is a trigram representation instead of a single subword representation. As we see, self-attention can capture global features regardless of distance. We take the first token `[CLS]` as an example, and this self-attention assigns high weights for the token `e` and `[SEP]`,

which constructs a representation like this: `[CLS]b + me[SUB] + ##me[SEP]`. This segment tells us this word starts with `b` and ends with `me`.

5.6 Conclusion

We have presented a lightweight contrastive-learning framework, LOVE, to learn word representations that are robust even in the face of out-of-vocabulary words. Through a series of empirical studies, we have shown that our model (with only 6.5M parameters) can achieve similar or even better word embeddings on both intrinsic and extrinsic evaluations compared to other mimick-like models. Moreover, our model can be added to models with static embeddings (such as FastText) or dynamical embeddings (such as BERT) in a plug-and-play fashion, and bring significant improvements there. For future work, we aim to extend our model to languages other than English.

6 Robustness: A Weakness of Positional Encodings

Positional Encodings (PEs) are used to inject word-order information into transformer-based language models. While they can significantly enhance the quality of sentence representations, their specific contribution to language models are not fully understood, especially given recent findings that building natural-language understanding from language models with positional encodings are insensitive to word order. In this work, we conduct more in-depth and systematic studies of positional encodings, thus complementing existing work in two aspects: (1) We uncover the core function of PEs by identifying two common properties, Locality and Symmetry; (2) We first point out a potential weakness of current PEs by introducing two new probing tasks of word swap. We hope these new probing results and findings can shed light on how to design and inject positional encodings into language models.

6.1 Introduction

Transformer-based language models with Positional Encodings (PEs) can improve performance considerably across a wide range of natural language understanding tasks. Existing work resort to either fixed [270, 253, 212] or learned [239, 60, 277] PEs to infuse order information into attention-based models. To understand how PEs capture word order, prior studies apply visualized [283] and quantitative analyses [276] to various PEs, and their findings conclude that all encodings, both human-designed and learned, exhibit a consistent behavior: First, the position-wise weight matrices show that non-zero values gather on local adjacent positions. Second, the matrices are highly symmetrical, as shown in Figure 6.1. These are intriguing phenomena, with reasons not well understood.

To bridge this gap, we strive to uncover the core properties of PEs by introducing two quantitative metrics, *Locality* and *Symmetry*. Our empirical studies demonstrate that these two properties are correlated with sentence representation capability. This explains why fixed encodings are designed to satisfy them and learned encodings are

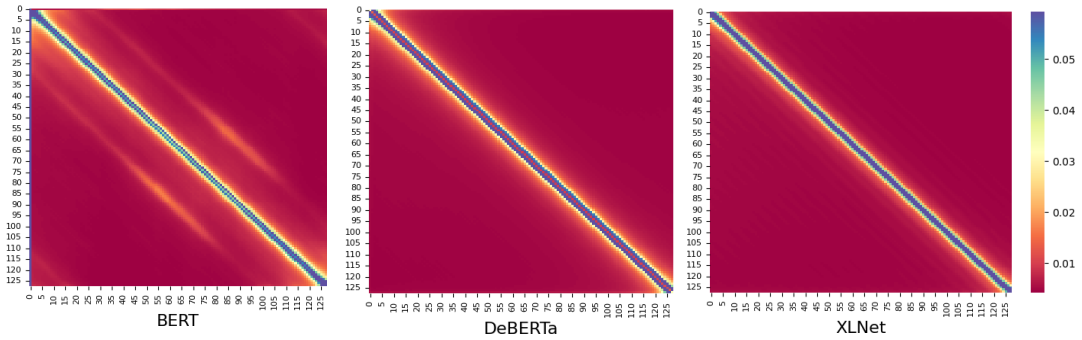

Figure 6.1: Visualizations of different pre-trained language models by using Identical Word Probing [276]. The attention weights are averaged across different layers.

favorable to be local and symmetrical. Moreover, we show that if BERT is initialized with PEs that already share good locality and symmetry, it can obtain better inductive bias and significant improvements across 10 downstream tasks.

Although PEs with locality and symmetry can achieve promising results on natural language understanding tasks (such as GLUE [275]), the symmetry property itself has an obvious weakness, which is not revealed by previous work. Existing studies use shuffled text to probe the sensitivity of PEs to word orders [298, 208, 247, 97, 1], and they all assume that the meaning of sentences with random swaps remains unchanged. However, the random shuffling of words may change the semantics of the original sentence and thus cause the change of labels. For example, the sentence pair below from SNLI [23] satisfies the entailment relation:

a. *A man playing an electric guitar on stage* b. *A man playing guitar on stage*

If we change the word order of the premise sentence so that it becomes "*an electric guitar playing a man on stage*", but a fine-tuned BERT still finds that the premise entails the hypothesis. Starting from this point, we design two new probing tasks of word swap: *Constituency Shuffling* and *Semantic Role Shuffling*. The former attempt to preserve the original semantics of the sentence by swapping words inside constituents (local structure) while the latter intentionally changes the semantics by swapping the semantic roles in a sentence (global structure), e.g., the agent and patient. Our probing results show that existing language models with various PEs are robust against local swaps but extremely fragile against global swaps.

6.2 Preliminaries

The central building block of transformer architectures is the self-attention mechanism [270]. Given an input sentence: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^{n \times d}$, where n is the number of words and d is the dimension of word embeddings, then the attention computes the output of the i-th token in this way:

$$\bar{\mathbf{x}}_i = \sum_{j=1}^{n} \frac{\exp(\alpha_{i,j})}{Z} \mathbf{x}_j \mathbf{W}^V \qquad \text{where} \quad \alpha_{i,j} = \frac{(\mathbf{x}_i \mathbf{W}^Q)(\mathbf{x}_j \mathbf{W}^K)^\top}{\sqrt{d}}, Z = \sum_{j=1}^{n} \exp(\alpha_{i,j}) \tag{6.1}$$

Self-attention heads do not intrinsically capture the word orders in a sequence. Therefore, specific methods are used to infuse positional information into self-attention [65].

Absolute Positional Encoding (APE) computes a positional encoding for each token and add it to the input content embedding to inject position information in the original sequence. The $\alpha_{i,j}$ in Equation 6.1 are then written:

$$\alpha_{i,j} = \frac{(\mathbf{x}_i + \mathbf{p}_i)\mathbf{W}^Q\big((\mathbf{x}_j + \mathbf{p}_j)\mathbf{W}^K\big)^\top}{\sqrt{d}} \tag{6.2}$$

where $\mathbf{p}_i \in \mathbb{R}^d$ is a position embedding of the ith token, obtained by **fixed** [270, 58, 263, 245, 253] or **learned** encodings [84, 60, 277, 213]. Further, TUPE model simplifies Equation 6.2 by removing two redundant items:

$$\alpha_{i,j} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\top + (\mathbf{p}_i\mathbf{U}^Q)(\mathbf{p}_j\mathbf{U}^K)^\top}{\sqrt{d}} \tag{6.3}$$

In absolute positional encoding, the equation 6.2 can be expanded as:

$$\alpha_{ij} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\top}{\sqrt{d}} + \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{p}_j\mathbf{W}^K)^\top}{\sqrt{d}} + \frac{(\mathbf{p}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\top}{\sqrt{d}} + \frac{(\mathbf{p}_i\mathbf{W}^Q)(\mathbf{p}_j\mathbf{W}^K)^\top}{\sqrt{d}} \tag{6.4}$$

There are four terms in this expression: context-to-context, context-to-position, position-to-context, and position-to-position. While the first and the fourth term are intuitive, the token encodings and positional encodings do not have strong correlations with each other, and the context-position correlations may even induce unnecessary noise. Based on this analysis proposed by TUPE (Transformer with Untied Positional Encoding) [135] that removes the second and third redundant terms and introduces different parameters for the position encoding:

$$\alpha_{ij} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\top + (\mathbf{p}_i\mathbf{U}^Q)(\mathbf{p}_j\mathbf{U}^K)^\top}{\sqrt{d}}, \tag{6.5}$$

Here, $\mathbf{U}^Q$ and $\mathbf{U}^K$ are weights that need to be learned, capturing positional queries and keys, respectively. Their empirical results confirm that the removal of the two context-to-position terms consistently improves the model performance on a series of tasks.

Relative Positional Encoding (RPE) produces a vector $\mathbf{r}_{i,j}$ or a scalar value $\beta_{i,j}$ that depends on the relative distance of tokens. Specifically, these methods apply such vector or bias to the attention head so that the corresponding attentional weight can be updated based on the relative distance of two tokens [239, 221]:

$$\alpha_{i,j} = \frac{\mathbf{x}_i\mathbf{W}^Q\big(\mathbf{x}_j\mathbf{W}^K + \mathbf{r}^K_{i,j}\big)^\top}{\sqrt{d}} \quad ; \quad \alpha_{i,j} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\top + \beta_{i,j}}{\sqrt{d}} \tag{6.6}$$

where the left mode uses a vector $\mathbf{r}_{i,j}$ while the right uses a scalar value $\beta_{i,j}$, for infusing relative distance into attentional weight.

Recent research of RPEs has been remarkably vibrant, with the emergence of diverse novel and promising variants [54, 106, 212].

Unified Positional Encoding. Inspired by TUPE [135], we rewrite all above absolute and relative positional encodings in a unified way:

$$\alpha_{i,j} = \frac{\overbrace{\gamma_{i,j}}^{contextual} + \overbrace{\delta_{i,j}}^{positional}}{\sqrt{d}} \tag{6.7}$$

where, the left half of the numerator, $\gamma_{i,j}$, captures contextual correlations (or weights), i.e., the semantic relations between token x_i and x_j. In this case, it is $\gamma_{i,j} = (\mathbf{x}_i \mathbf{W}^Q)(\mathbf{x}_j \mathbf{W}^K)^\top$. δ, the right half, captures positional correlations, i.e., the positional relations between tokens x_i and x_j. For example, TUPE's positional correlation can be represented as $\delta_{i,j} = (\mathbf{p}_i \mathbf{W}^Q)(\mathbf{p}_j \mathbf{W}^K)^\top$ while the relative encoding in [239] can be represented as $\delta_{i,j} = \mathbf{x}_i \mathbf{W}^Q (\mathbf{r}^K_{i,j})^\top$. Thus, existing positional encodings all add contextual and positional correlations together in every attention head.

6.3 Positional Encodings Enforce Locality and Symmetry

6.3.1 The Properties of Locality and Symmetry

Existing work analyze positional encodings with the help of visualizations [283, 276, 1], and their analyses of either fixed or learned encodings led to similar visualized results, as shown in Figure 6.1. These position-wise weight matrices are computed by using the *Identical Word Probing* [276]: many repeated identical words are fed to the pre-trained language model, so that the attention values ($\alpha_{i,j}$) in Equation 6.7) are unaffected by contextual weights. Each matrix in this figure is a positional weight map, where each row is a vector for the i-th position and the element at $(i,\ j)$ indicates the attention weight between the i-th position and the j-th position. We can first observe these attention matrices are all diagonal heavy, which means various positional encodings highly attend to local positions. Second, all matrices are nearly symmetrical. We call these two phenomena the *Locality* and *Symmetry* of positional encodings. The symmetry property has been discovered and quantified already by prior work [276]. Here, we provide a more in-depth analysis of symmetry. We will also point out the potential flaw of symmetry itself, which is not considered by prior work. To better understand how encodings capture word order, we introduce two quantitative metrics to depict the Locality and Symmetry for an attentional weight vector $\boldsymbol{\varepsilon}_\mathbf{i}$, where the element $\varepsilon_{i,j}$ can be denoted as:

$$\varepsilon_{i,j} = \frac{\exp(\alpha_{i,j})}{\sum_{j=1}^{n} \exp(\alpha_{i,j})} \qquad \text{where} \quad \varepsilon_{i,j} \geq 0 \quad \text{and} \quad \sum_{j=1}^{n} \varepsilon_{i,j} = 1 \tag{6.8}$$

Locality is a metric that depicts the degree of the gathering of weights in local positions for an attentional weight vector. Given a weight vector for the i-th position $\boldsymbol{\varepsilon}_\mathbf{i} = \{\varepsilon_{i,1}, \varepsilon_{i,2}, ..., \varepsilon_{i,n}\}$, we define locality as:

$$\text{Locality}(\boldsymbol{\varepsilon}_\mathbf{i}) \in [0,1] = \sum_{j=1}^{n} \frac{\varepsilon_{i,j}}{2^{|i-j|}} \tag{6.9}$$

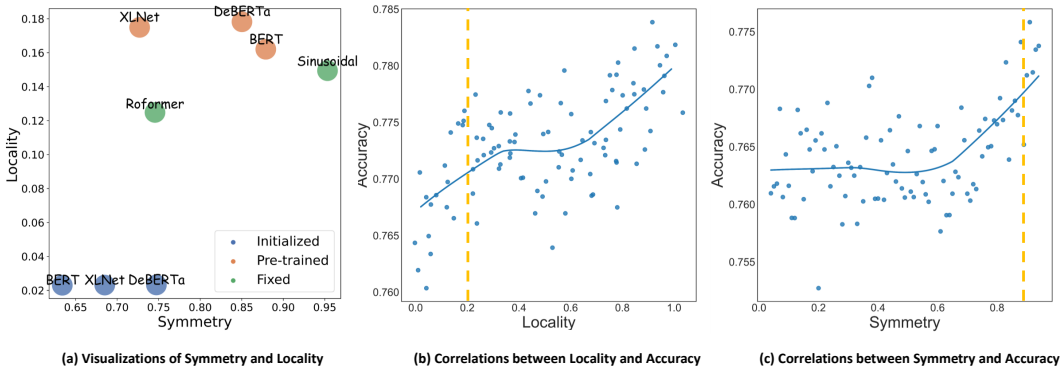

Figure 6.2: Empirical studies of the properties of locality and symmetry. The accuracy is tested on the MR dataset [199] The yellow line shows the locality or symmetry for the pre-trained BERT.

Here, a value of 1 means the vector perfectly satisfies the locality property. For example, given a sequence whose length is 5 and a weight vector for the first position $[1,0,0,0,0]$, the locality is 1, which means it perfectly matches the locality. In contrast, the locality is $1/16$ if the weight only attends the last position $[0,0,0,0,1]$. For measuring the locality of a matrix, we average the locality values of all vectors in the matrix. ***Symmetry*** is a metric that describes how symmetrical the weights scatter around the current position for an attentional weight vector. We adapt the *Symmetrical Discrepancy* [276] for this goal:

$$\text{Symmetry}(\varepsilon_i) \in [0,1] = 1 - \sum_{j=1}^{\lfloor n/2 \rfloor} \text{Norm}\left(\frac{\left|\varepsilon_{i,j} - \varepsilon_{i,n-j+1}\right|}{\lfloor n/2 \rfloor}\right) \tag{6.10}$$

Here, a value of 1 means that the vector is completely symmetrical. We modify the original formula in two points: First, we apply a min-max normalization to each position to obtain more uniform distributions, because the values of the original one extremely cluster around 0. Second, we reverse the value so that 1 means a perfect symmetry instead of 0. Likewise, the average value of all vectors in a matrix is used as the matrix-level symmetry.

6.3.2 Are Locality and Symmetry Learned?

The manually designed encodings Sinusoidal [270] and Roformer [253] both satisfy the symmetry and locality properties. However, it is not clear why they were designed this way. More surprisingly, learned encodings all display locality and symmetry. Therefore, one may ask whether the two properties are learned after pre-training, and what effect they have.

To answer this question, we use our two proposed metrics to quantify the positional weight matrix (the averaged weight across layers) before and after pre-training. Specifically, three language models, BERT [60], XLNet [300] and DeBERTa [106] are tested in this experiment. As shown in the left in Figure 6.2, the three language models all become much more local and symmetrical after pre-training, which proves that the two properties are indeed learned.

To further explain why positional encodings have a preference for learning these two properties, we probe the correlations between the two properties and the representation ability in downstream tasks. To avoid pre-training all language models from scratch, we use static word embeddings from GloVe [204] and an encoder that is fully based on our handcrafted positional encodings for a sentence classification task. The benefit is that we can adjust the hyper-parameter in the handcrafted encodings to obtain encodings with different degrees of locality and symmetry, so that we can evaluate the correlations precisely. Specifically, we obtain around 100 encoders whose locality (or symmetry) varies from 0.01 to 1.0 and test their accuracy on the MR sentiment analysis task. We will describe our handcrafted encodings in Section 6.3.3.

The middle figure in Figure 6.2 shows the results for different locality values. In this experiment, the symmetry value is 1.0 for all encoders. We observe that the accuracy constantly increases as the locality of encodings strengthens, which means a higher locality induces better sentence representation. The yellow line is the locality value for BERT (around 0.2), and BERT actually does not have an extreme locality, which means that a perfect locality is unnecessary. The right figure in Figure 6.2 shows the results for different symmetry values. In this experiment, we vary the symmetry while keeping the locality in the interval $[0.15, 0.3]$, which is close to the value of BERT. Because the change of symmetry will impact the value of locality, we can only observe this type of partial correlation. We find that symmetry affects performance only after a certain value (0.65), and a better symmetry leads to better accuracy. Also, the encodings of the pre-trained BERT are highly symmetrical.

We conclude that positional encodings with more suitable locality and symmetry can yield better performance on downstream tasks, which may explain why fixed encodings are designed to meet the two properties and why learned encodings all exhibit this behavior. However, encodings are not perfectly local, which might be due to the network architectures and the specific target tasks.

6.3.3 Can Locality and Symmetry Yield Better Inductive Bias?

Given that locality and symmetry stand out as important learned features of existing positional encodings, it begs the question that what happens if a language model is initialized with positional encodings with good locality and symmetry.

For this purpose, we replace the positional correlations $\delta_{i,j}$ in Equation 6.7 with handcrafted Positional Encodings to probe. There are various human-designed positional encodings, e.g., sinusoidal encodings [270], rotary encodings [253] and ALiBi [212], but the locality and symmetry cannot be modified easily for these encodings. To address this issue, we propose the *Attenuated Encoding*, which use a Gaussian kernel [95]:

$$\delta_{i,j} = \Phi(l_{i,j}) = \frac{\exp(\alpha_{i,j})}{\sum_{j=1}^{n} \exp(\alpha_{i,j})} \qquad \text{where} \quad \alpha_{i,j} = \begin{cases} -s\,w\,l_{i,j}^2 & i \le j \\ -w\,l_{i,j}^2 & i > j \end{cases} \tag{6.11}$$

where $l_{i,j}$ is the relative distance, $w > 0$ is a scalar parameter that controls the locality value, and s is a scalar parameter that controls the symmetry value. Note that there

Model	Size	Sentiment Analysis			Textual Entailment			Paraphrase Identification		Textual Similarity		Avg
		MR (22K)	SUBJ (20K)	SST-2 (68.8K)	QNLI (110K)	RTE (5.5K)	MNLI (413K)	MRPC (5.4K)	QQP (755k)	STS-B (8.4K)	SICK-R (9.4K)	
BERT	110M	$72.5_{\pm 5.3}$	$91.0_{\pm 2.7}$	$86.4_{\pm 2.7}$	$85.8_{\pm 1.0}$	$59.2_{\pm 1.2}$	$78.2_{\pm 0.8}$	$73.5_{\pm 1.8}$	$88.7_{\pm 0.6}$	$77.8_{\pm 4.1}$	$64.9_{\pm 6.0}$	77.8
BERT-A^*-s	113M	$79.4_{\pm 2.9}$	$93.7_{\pm 0.6}$	$88.0_{\pm 0.7}$	$86.3_{\pm 1.1}$	$59.4_{\pm 2.7}$	$78.8_{\pm 0.4}$	$81.5_{\pm 2.2}$	$88.7_{\pm 0.4}$	$83.6_{\pm 2.0}$	$76.3_{\pm 1.1}$	81.6
BERT-A^*	138M	$78.2_{\pm 3.5}$	$93.0_{\pm 0.8}$	$88.1_{\pm 1.0}$	$87.0_{\pm 0.5}$	$61.0_{\pm 1.4}$	$78.9_{\pm 0.9}$	$80.9_{\pm 3.9}$	$89.2_{\pm 0.3}$	$84.3_{\pm 2.5}$	$76.0_{\pm 4.7}$	81.7

Table 6.1: Evaluations of handcrafted encodings across 10 downstream tasks. We report the average score (Spearman correlation for textual similarity and accuracy for others) of five runs using different learning rates. * means the encodings are learnable and s means that positional encodings are shared within the attention headers of layers.

are two key differences between our encodings and other manually designed ones such as the T5 bias [221] and ALiBi [212]. First, the output generated by our method is an attentional vector (or a discrete probability distribution) that can be regarded as a type of attention mechanism. Thus, we can estimate the locality and symmetry individually. ALiBi biases, in contrast, cannot be measured by our proposed metrics directly. Second, we can adjust the hyper-parameters in our method for obtaining encodings with different localities and symmetry, which ALiBi does not allow.

In this experiment, we adjust the parameter w and s for obtaining weight vector δ that share similar locality and symmetry with pre-trained BERT (Locality=0.17 and Symmetry=1.0). After, we pre-train BERT$_{base}$ initialized with δ and compare them to learned encodings on downstream natural language understanding tasks. Three variants are compared with the original BERT: 1) BERT-A^*-s uses learnable and shared δ, but the weights are shared inside a particular layer; 2) BERT-A^* uses learnable but not shared δ, which means δ is different in each attentional head. The empirical results are shown in Table 6.1. We observe that both BERT-A^*-s and BERT-A^* can significantly outperform the original BERT, which demonstrates positional encodings with initialization of suitable locality and symmetry can have better inductive bias in sentence representation.

6.3.4 What Is the Drawback of Symmetry?

Although positional encodings with good symmetry perform well on a series of downstream tasks, the symmetry property has an obvious flaw in sentence representations, which is ignored by prior studies. Existing probes study the sensitivity of language models to word order by shuffling the words in a sentence, and they can be roughly divided into three categories: random swap [208, 97, 1], n-gram swap [247], and subword-level phrase swap [50]. All these works assume that the labels of the randomly shuffled sentences are unchanged. However, this is obviously not the case. In particular, the shuffled sentence may have another label (think of the textual entailment example from the introduction).

To address the issue, we propose two new probing tasks of word swaps: *Constituency Shuffling* and *Semantic Role Shuffling*. *Constituency Shuffling* aims to disrupt the inside order of constituents, which is able to change the word order while preserving

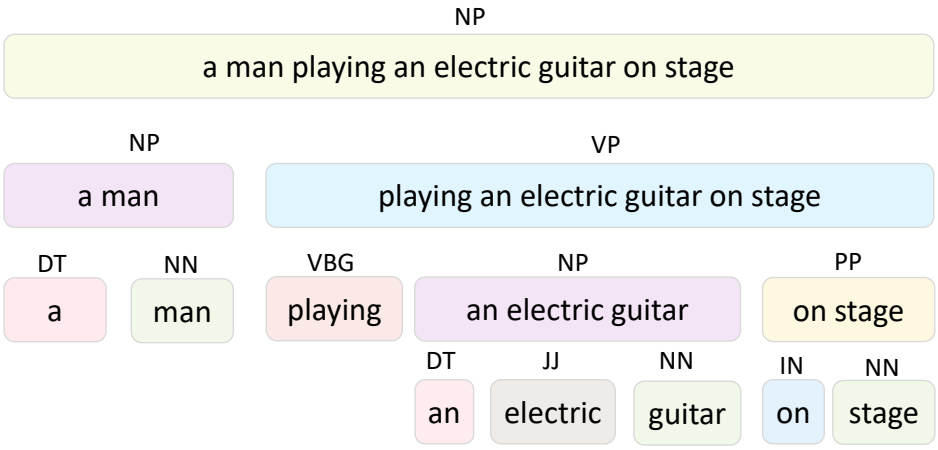

Figure 6.3: Illustration of constituent parsing for one sentence in SNLI "*a man playing an electric guitar on stage*". The result is generated by Berkeley Neural Parser.

	Original	Shuffled
Shuffled-3	*An old man with a package poses in front of an advertisement .*	*An man old with package a poses in front of advertisement an .*
Shuffled-4	A land rover is being driven across a river .	A land rover is being a driven river across .
Shuffled-5	A man reads the paper in a bar with green lighting .	A man reads the paper in with green a lighting bar .
Shuffled-6	A little boy in a gray and white striped sweater and tan pants is playing on a piece of playground equipment .	A little boy in striped a sweater and white gray and tan pants is playing piece playground of equipment on a .
Shuffled-SR	several women are playing volleyball .	volleyball are playing several women .
Shuffled-SR	a man and woman are sharing a hotdog .	a hotdog are sharing a man and woman .

Table 6.2: Some cases of the shuffled SNLI datasets in our word swap probing. Texts in the same color mean the corresponding phrases.

the maximum degree of original semantics. A constituent parsing case is shown in Figure 6.3, and we can shuffle the word order inside some phrases, e.g., the noun phrase "*an electric guitar*" while the semantic will not be changed (the grammar structure may be destroyed). We construct different shuffled datasets by phrase length, e.g., "*an electric guitar*" is a phrase of length 3 and we can obtain tri-gram shuffled sets. Datasets constructed by constituency shuffling are referred to as Shuffle-*x* and *x* means the length of phrase. On the other hand, *Semantic Role Shuffling* intentionally changes the semantics by swapping the order of the agent and patient of sentences and thus results in a new sentence with different meanings. In Figure 6.3, "*a man*" as the entity that performs the action, technically known as the agent, and "*an electric guitar*" as the entity that is involved in or affected by the action, which is called the patient. We refer to this dataset as Shuffle-SR because it swap the semantic roles in a sentence. Some shuffled examples are shown in Table 6.2.

Model	Symmetry	Locality	Original	Shuffle-3 (Δ)	Shuffle-4 (Δ)	Shuffle-5 (Δ)	Shuffle-6 (Δ)	Random (Δ)	Original	Shuffle-SR (Δ)
BERT	87.9	16.2	89.8	-0.4	-0.6	-0.3	-0.5	-2.7	89.8	-63.9
ALBERT	82.0	20.3	91.8	-0.5	-1.1	-1.3	-1.7	-6.0	92.0	-66.8
DeBERTa	85.0	17.8	91.6	-0.5	-0.7	-1.3	-1.1	-5.1	91.6	-58.9
XLNet	72.7	17.5	91.5	-0.2	-0.3	-0.7	-1.2	-5.4	91.3	-57.8
StrucBERT	96.3	7.5	90.9	-0.5	-0.9	-1.3	-1.1	-4.4	90.8	-44.6

Table 6.3: Results of Constituency Shuffling and Semantic Role Shuffling, measured by accuracy. Shuffle-*x* means phrases with length *x* are shuffled. Shuffle-SR means the semantic roles of agent and patient are swapped.

The distinction of our proposed two probing tasks is that one preserves the semantics while another changes the semantics. Then, we can probe the capability of language models to correctly recognize the new sentence's meaning. Specifically, the Stanford Natural Language Inference (SNLI) [23] dataset is used in this experiment and it provided constituent structure for each sentence. To probe the sensitivity of language models to the two types of shuffling, we fine-tune 5 pre-trained language models with good symmetry on SNLI training set and evaluate them on the newly constructed Shuffle-*x* and Shuffle-SR datasets. The overall results of word swap probing are shown in Table 6.3 We first observe performances of all language models across Shuffle-*x* sets basically do not degenerate, which confirms the benefits of the locality and symmetry properties. Second, most models fail on the Shuffle-SR dataset, which demonstrates local symmetry does not capture global position changes well, which explain the reason that BERT fails on the example: "*an electric guitar playing a man on stage*". Although the local symmetry learned by positional encodings can performs well on a series of language understanding tasks, the symmetry itself has obvious flaws. The better performance of StrucBERT on the Shuffle-SR suggests that introducing additional order-sensitive training tasks may improve this problem.

6.4 Conclusion

We have proposed a series of probing analyses for understanding the role of positional encodings in sentence representations. We find two main properties of existing encodings, Locality and Symmetry, which is correlated with the performance of downstream tasks. Meanwhile, we point out an obvious flaw of the symmetry property.

The limitations of this work are two-fold. First, our analysis is limited to the natural language understanding of the English language. Different languages display different word order properties. For instance, English is subject-verb-object order (SVO) while Japanese is subject-object-verb order (SOV), and natural language generation models are not included in this work. Besides, a recent work finds that the autoregressive models without any explicit positional encoding are still competitive with standard models [103], which shows the generative model might not be a perfect target for researching order information. Second, although our handcrafted positional encodings satisfy the Symmetry property, it merely replicates the limitations of current positional encoding, albeit in a simplified form. Further architecture development should address the problem of the "*an electric guitar playing a man on stage.*" mentioned in the

introduction.

7 Application: Using Entity Disambiguation Models for Knowledge Base Completion

Despite their impressive scale, knowledge bases (KBs), such as Wikidata, still contain significant gaps. Language models (LMs) have been proposed as a source for filling these gaps. However, prior works have focused on prominent entities with rich coverage by LMs, neglecting the crucial case of long-tail entities. In this paper, we present a novel method for LM-based-KB completion that is specifically geared for facts about long-tail entities. The method leverages two different LMs in two stages: for candidate retrieval and for candidate verification and disambiguation. To evaluate our method and various baselines, we introduce a novel dataset, called MALT, rooted in Wikidata. Our method outperforms all baselines in F1, with major gains especially in recall.

7.1 Introduction

Knowledge base completion (KBC) is crucial to continuously enhance the scope and scale of large knowledge graphs (KGs). It is often cast into a link prediction task: infer an O(bject) argument for a given S(ubject)-P(redicate) pair. However, the task is focused on the KG itself as the only input, and thus largely bound to predict SPO facts that are also derivable by simple logical rules for inverse predicates, transitive predicates etc. [5, 259]. To obtain truly new facts, more recent methods tap into large language models (LMs) that are learned from huge text collections, including all Wikipedia articles, news articles and more. The most promising approaches to this end generate cloze questions for knowledge acquisition and ask LMs to generate answers [207]. The LM input is often augmented with carefully crafted short prompts (e.g., a relevant Wikipedia paragraph) [244, 124, 217].

However, notwithstanding great success for question answering to humans, the LM-based approach falls short on meeting the high quality requirements for enriching a KG with crisp SPO facts. Even if most answers are correct, there is a non-negligible fraction of false or even "hallucinated" outputs by the LM, and large KGs, like Wikidata [274], cannot tolerate error rates above 10 percent. Moreover, even correct answers are not

properly canonicalized: they are surface phrases and not unique entities in the KG. These problems are further aggravated when the to-be-inferred O arguments are *long-tail* entities, with very few facts in Wikidata. Here, we call an entity *long-tail* when it has less than 14 triples in Wikidata, because nearly 50% of the Wikidata entities have fewer than 14 triples. These are exactly the pain point that calls for KBC. This paper addresses this problem.

As an example, consider the late Canadian singer *Lhasa de Sela*. Wikidata solely covers basic biographic facts and selected awards, nothing about her music. However, text sources such as her Wikipedia article or other web pages provide expressive statements about her albums, songs, collaborations etc. For example, we would like to spot the facts that ⟨*Lhasa de Sela, collaboratedWith, Bratsch*⟩ and ⟨*Lhasa de Sela, performedSong, Anyone and Everyone*⟩. Note that capturing these as SPO facts faces the challenge of having to capture and disambiguate multi-word names (*"Lhasa de Sela"*) and common-noun phrases (*"anyone and everyone"*). When trying to extract such statements via cloze questions or more refined prompts to LMs such as GPT-3 [26] or chatGPT, the outputs would often be *"Lhasa"*, which is highly ambiguous, or *"everyone"*, which is incomplete and impossible to interpret.

Approach and Contribution. This paper devises a novel method for knowledge base completion (KBC), specifically geared to cope with long-tail entities. Although we will present experimental comparisons to prior works on relation extraction from text, we believe that ours is among the first works to successfully cope with the challenge of noise and ambiguity in the long tail.

Our method leverages Transformer-based language models in a new way. Most notably, we employ two different LMs in a two-stage pipeline. The first stage generates candidate answers to input prompts and gives cues to retrieve informative sentences from Wikipedia and other sources. The second stage validates (or falsifies) the candidates and disambiguates the retained answer strings onto entities in the underlying KG (e.g., mapping *"Lhasa"* to `Lhasa de Sela` , and "Bratsch" to `Bratsch (band)`).

The novel contributions of this work are the following:

- the first KBC method that leverages LMs to cope with long-tail entities;
- a new dataset, called MALT, to benchmark methods with long-tail entities;
- experimental comparisons with baselines, using the MALT data.

7.2 Related Work

Knowledge Base Completion. This task, KBC for short, has mostly been tackled as a form of link prediction: given a head entity S and a relation P, predict the respective tail entity O, using the KG as sole input. A rich suite of methods have been developed for this task, typically based on latent embeddings computed via matrix or tensor factorization, neural auto-encoders, graph neural networks, and more (see, e.g., surveys [44, 121] and original references given there). However, the premise of inferring missing facts from the KG itself is a fundamental limitation. Indeed, several studies have found that many facts predicted via the above KBC techniques are fairly obvious

and could also be derived by simple rules for transitivity, inverse relations etc. [5, 259].

Language Models as Knowledge Bases. The LAMA project [207] posed the hypothesis that probing LMs with cloze questions is a powerful way of extracting structured facts from the latently represented corpus on which the LM was trained. A suite of follow-up works pursued this theme further and devised improvements and extensions (e.g., [107, 123, 134, 227, 244, 313]). This gave rise to the notion of "prompt engineering" for all kinds of NLP tasks [167]. In parallel, other works studied biases and limitations of the LM-as-KB paradigm (e.g., [30, 69, 226, 124]). In this work, we investigate the feasibility of leveraging LMs to complete real-world KBs, and mainly focus on long-tail facts.

7.3 Two-Stage KBC Method

We propose an unsupervised method for KBC that taps into LMs as latent source for facts that cannot be inferred from the KG itself. Our method operates in two stages:

1. For a given S-P pair, generate candidate facts ⟨S,P,"O"⟩ where "O" is an entity name and possibly a multi-word phrase.
2. Corroborate the candidates, retaining the ones with high confidence of being correct, and disambiguate the "O" argument into a KG entity.

Candidate Generation. We devise a generic prompt template for cloze questions, in order to infer an "O" answer for a given S-P pair. This merely requires a simple verbalizer for the relation P:

"⟨S-type⟩ S ⟨P-verb⟩ which ⟨O-type⟩?"

(e.g., "the song ⟨S⟩ is performed by which person?" for the predicate `performer`). The S-type and O-type are easily available by the predicate type-signature from the KG schema. As additional context we feed a Wikipedia sentence from the S entity's article into the LM. This is repeated for all sentences in the respective Wikipedia article. Specifically, we employ the SpanBERT language model [130], which is fine-tuned on on the SQuAD 2.0 [223] [1]. Note that all of this is completely unsupervised: there is no need for any fine-tuning of the LM, and there is no prompt engineering.

Candidate Corroboration and Canonicalization. The first stage yields a scored list of candidates in the form of pairs ("O", *s*) with an entity name and a Wikipedia sentence *s*. In the corroboration stage, the candidates are fed into a second LM for re-ranking and pruning false positives. Specifically, we employ the generative entity disambiguation model GENRE [57], which in turn is based on BART [155] and fine-tuned on BLINK [291] and AIDA [113]. We construct the input by the template:

"⟨S-type⟩ S ⟨P-verb⟩ [ENT] this ⟨O-type⟩ [ENT]"

(e.g., "the song Anyone and Everyone is performed by [ENT] this person [ENT]"),

[1] https://huggingface.co/mrm8488/spanbert-large-finetuned-squadv2

Dataset	SPO triples	Long-tail fraction
DocRED [301]	63K	32.0 %
LAMA-TREx [207]	34K	39.6 %
X-FACTR [123]	46K	49.6 %
MALT (Ours)	49K	87.0 %

Table 7.1: Estimated fractions of long-tail S entities across different datasets, where long-tail means at most 13 triples in Wikidata. The estimations are based on 200 samples across 8 relations.

contextualized with the sentence *s*. GENRE generates a list of answer entities `E`, taken from an underlying KG, like Wikidata, that is, no longer just surface names. If the candidate name "O" approximately matches a generated `E` (considering alias names provided by the KG), then the entire fact, now properly canonicalized, is kept. Since we may still retain multiple facts for the same S-P input and cannot perfectly prevent false positives, the inferred facts are scored by an average of the scores from stage 1 and stage 2.

7.4 MALT: New Dataset for Benchmarking

Benchmarks for KBC and LM-as-KB cover facts for all kinds of entities, but tend to focus on prominent ones with frequent mentions. Likewise, benchmarks for relation extraction (RE) from text, most notably TACRED [308], DocRED [301] and LAMA [207] do not reflect the difficulty of coping with long-tail entities and the amplified issue of surface-name ambiguity (see Table 7.1. Therefore, we developed a new dataset with emphasis on the long-tail challenge, called MALT (for "**M**ulti-token, **A**mbiguous, **L**ong-**T**ailed facts").

To construct the dataset, we focus on three types of entities: `Business`, `MusicComposition` and `Human`, richly covered in Wikidata and often involving long-tail entities. We randomly select subjects from the respective relations in Wikidata, and keep all objects for them. We select a total of 8 predicates for the 3 types; Table 7.2 lists these and gives statistics.

The dataset contains 65.3% triple facts where the O entity is a multi-word phrase, and 58.6% ambiguous facts where the S or O entities share identical alias names in Wikidata. For example, the two ambiguous entities ,*"Birmingham, West Midlands (Q2256)"* and *"Birmingham, Alabama (Q79867)"*, have the same `Label` value *"Birmingham"*. In total, 87.0% of the sample facts have S entities in the long tail, where we define long-tail entities to have at most 13 Wikidata triples.

Subject Type	Relation	Wikidata ID	Triples	multi-token (%)	ambiguous (%)	long-tail (%)
Business	founded by	P112	5720	97.3	21.1	91.2
MusicComposition	performer	P175	1876	91.1	62.0	47.3
	composer	P86	3016	98.2	59.8	88.5
Human	place of birth	P19	13416	23.6	81.6	99.3
	place of death	P20	7247	25.9	84.8	99.6
	employer	P108	3503	96.5	37.4	81.4
	educated at	P69	13386	99.6	38.7	72.2
	residence	P551	886	32.1	87.1	96.4
Micro-Avg	-	-	-	65.3	58.6	87.0

Table 7.2: Statistics for MALT dataset.

Relation	ID	NER + RC (CNN)			REBEL			KnowGL			GenIE			Ours		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
founded by	P112	13.5	21.2	16.5	42.8	27.3	33.3	0.0	0.0	0.0	59.1	7.9	13.9	57.0	44.5	50.0
performer	P175	5.2	10.1	6.9	25.3	28.1	26.6	0.0	0.0	0.0	47.3	19.1	27.2	42.7	15.6	22.9
composer	P86	17.3	20.5	18.8	37.9	27.7	32.0	37.6	25.7	30.6	70.0	16.6	26.8	67.3	65.6	66.4
place of birth	P19	4.7	4.7	4.7	49.3	20.5	28.9	49.4	23.4	31.7	64.1	9.2	16.1	47.9	61.4	53.8
place of death	P20	12.5	4.7	6.8	52.6	11.8	19.2	66.6	9.4	16.5	47.5	3.0	5.6	46.6	48.2	47.4
employer	P108	8.7	4.9	6.3	50.0	4.9	8.8	0.0	0.0	0.0	54.0	0.1	0.2	30.0	29.3	29.6
educated at	P69	8.9	8.4	7.7	15.4	1.1	2.1	22.2	1.1	2.2	46.7	0.1	0.2	42.9	39.5	41.2
residence	P551	0.0	0.0	0.0	33.3	8.3	13.3	33.3	8.3	13.3	44.4	0.2	0.4	19.2	41.7	26.3
Micro-Avg	-	26.7	13.7	13.7	38.3	16.2	20.6	26.2	8.5	11.8	52.2	6.9	11.2	44.2	43.2	42.2

Table 7.3: Performance comparison on MALT data.

7.5 Experimental Evaluation

Baselines. To the best of our knowledge, there is no prior work on KBC or LM-as-KB that is specifically geared for coping with long-tail entities. As a proxy, we thus compare to several state-of-the-art methods for relation extraction (RE) from text. At test time, these methods receive the retrieved Wikipedia sentences for a ground-truth SPO fact and the SP pair as input, and are run to extract the withheld O argument (sentence-level extraction).

We compare to the following baselines:

- *NER + RC (CNN)* uses TNER [267] to recognize entity mentions in context sentences, followed by a CNN-based relation classifier **(author?)** [190]. The RC component is trained on REBEL [29].
- *REBEL* [29] is an end-to-end relation extraction for more than 200 different relation types in Wikidata.
- *KnowGL* [229] is an open-source system that can convert text into a set of Wikidata statements.
- *GenIE* [132] is an end-to-end closed triplet extraction model, which is trained on REBEL dataset [29]. GenIE uses Wikidata as the target KB and can extract 5,891,959 entities and 857 relations.

Setup. There are two hyper-parameters for all competitors, the number of candidates k (or the "top-k" hyper-parameter for baseline models) and the threshold α for cutting off the extracted triples. For our framework, k is 20 for all competitors and the threshold α is learned by using a hold-out (20%) validation set. We report results for precision, recall and F1, with the original Wikidata triples as ground truth. Although MALT provides canonicalized entities, we consider the extracted O to be a correct prediction as long as it appears in the alias table because some baselines themselves cannot do disambiguation.

Our method is completely unsupervised, and the only additional cost is prompt. We manually design one template for each relation (as shown in Table 7.4).

Relation	ID	Candidate Generation	Corroboration and Canonicalization
founded by	P112	the business `[x]` is founded by which person?	the business `[x]` is founded by `[ENT]` this person `[ENT]`
performer	P175	the song `[x]` is performed by which person?	the song `[x]` is performed by `[ENT]` this person `[ENT]`
composer	P86	the song `[x]` is composed by which person?	the song `[x]` is composed by `[ENT]` this person `[ENT]`
place of birth	P19	the person `[x]` was born in which place?	the person `[x]` was born in `[ENT]` this place `[ENT]`
place of death	P20	the person `[x]` died in which place?	the person `[x]` died in `[ENT]` this place `[ENT]`
employer	P108	the person `[x]` worked in which place?	the person `[x]` worked in `[ENT]` this place `[ENT]`
educated at	P69	the person `[x]` graduated from which place?	the person `[x]` graduated from `[ENT]` this place `[ENT]`
residence	P551	the person `[x]` lived in which place?	the person `[x]` lived in `[ENT]` this place `[ENT]`

Table 7.4: Prompts for relations in MALT. `[x]` is a placeholder for the subject entity and `[ENT]` is a special token for the mention.

Results. Table 7.3 shows the results from this experimental comparison. We observe that the GenIE baselines does well in terms of precision, but has very poor recall. In contrast, our two-stage method achieves both good precision and recall. Regarding precision, it is almost as good as GenIE (44% vs. 52%); regarding recall, it outperforms GenIE and the other baselines by a large margin (43% vs. 7%). Our method still leaves substantial room for further improvement, underlining the challenging nature of inferring facts for long-tail entities. We think of our method as a building block to aid a human curator by judicious suggestions for facts that would augment the KG.

Many of the inferred SPO facts are indeed completely missing in Wikidata; so they are also not in the withheld ground-truth samples for the above evaluation. To estimate how many facts we could potentially add to the KG and how good our automatically inferred predictions are, we picked 25 samples for each relation, a total of 250 fact candidates, and asked human annotators to assess their correctness. Over all relations, this achieved an average precision of 61%.

For the relation `educated at`, our method even has 76% precision, and this is a

case where the KG has enormous gaps: out of 10M sampled entities of type `Human`, only 65% have facts for this relation. For this case, our KBC method collected 1.2M candidate facts, showing the great potential towards closing these gaps.

7.6 Conclusion

We highlighted the challenge of knowledge base completion (KBC) for long-tail entities, introduced the MALT dataset for experimental comparisons and fostering further research, and presented a completely unsupervised method for augmenting knowledge bases with long-tail facts. Our method operates in two stages, candidate generation and candidate corroboration (incl. disambiguation), and leverages two different LMs in a complementary way. Experimental results show substantial gains over state-of-the-art baselines, and highlight the benefits of our two-stage design with two LMs complementing each other.

8 Conclusion

8.1 Summary

In this thesis, we addressed challenges related to entity disambiguation. More concretely, we worked on the following topics:

Biomedical Entity Disambiguation In chapter 3, we propose a lightweight yet effective neural network for biomedical entity disambiguation. Our model is 23 times smaller abd 6.4 times faster than the BERT-base model, and empirical results demonstrate that the model is very competitive, and achieves a performance that is statistically indistinguishable from the state of the art.

General Acronym Disambiguation In chapter 4, we present our constructed GLADIS, a new benchmark for acronym disambiguation, which is larger and more challenging than existing work. This benchmark contains three components: a larger dictionary, three datasets from the general, scientific, and biomedical domains, and a large-scale pre-training corpus. We have also proposed AcroBERT, the first pre-trained language model for acronym disambiguation, which can significantly outperform other baselines across multiple domains.

Out-of-Vocabulary Problem In chapter 5, we address the out-of-vocabulary by designing a contrastive learning framework, which named LOVE (Learning Out-of-Vocabulary Embeddings). LOVE is able to generate word representations for any unseen words by learning the behavior of pre-trained embeddings using only the surface form of words. LOVE can make language models more robust with few additional parameters. Extensive evaluations demonstrate that our lightweight model achieves similar or even better performances than prior competitors, both on original datasets and on corrupted variants.

Positional Encodings in Transformers In chapter 6, we point out one potential weakness of positional encodings, which are widely used in Transformer-based models.

Positional encodings are used to inject word-order features into language models. Although they can significantly enhance sentence representations, their specific function to language models are not fully understood, especially given recent findings that building natural-language understanding from language models with positional encodings are insensitive to word order. In this work, we conduct more in-depth and systematic studies of positional encodings, thus complementing existing work in two aspects: We first reveal the core function of PEs by identifying two common properties, Locality and Symmetry. After, we first point out a potential weakness of current PEs by introducing two new probing tasks of word swap. We hope these new probing results and findings can shed light on how to design and inject positional encodings into language models.

Knowledge Base Completion In chapter 7, we present a novel method for LM-based-KB completion that is specifically geared for facts about long-tail entities. Specifically, we present a novel method for LM-based-KB completion that is specifically geared for facts about long-tail entities. The method leverages two different LMs in two stages: for candidate retrieval and for candidate verification and disambiguation. To evaluate our method and various baselines, we introduce a novel dataset, called MALT, rooted in Wikidata. Our method outperforms all baselines in F1, with major gains especially in recall.

8.2 Future Work

There are still many challenges remaining in the task of entity disambiguation. We here provide some possible research directions that would be worth exploring in the future.

A Cleaner GLADIS We see one main limitations of the GLADIS benchmark. The current acronym dictionary is of relatively high quality, it still contains a small fraction of duplicate long forms due to typos (as in *"Convlutional Neural Network"*), morphological changes (as in *"Convolutional Neuronal Network"*) and nested acronyms (as in *"convolutional NN"*). A manual evaluation of 100 randomly chosen long forms from the three datasets in GLADIS shows that 6% of them are noisy. At the same time, the frequency of these noisy forms is much lower than that of the standard long forms: all noisy forms in the sample taken together appear 100 times in the corpus – compared to 31k times for the clean forms. Thus, the percentage of clean forms, weighted by their frequency, is 97%. A good AD system should select the most frequent one among noisy forms for an acronym, and in our sample none of the most frequent forms was noisy. One possible future work is to clean these noisy long forms in a manual or automatic way.

Phrase-LOVE The current LOVE focuses on word-level representations. In the future, the contrastive learning framework of LOVE might be extended to phrase-level representations. We envision the Phrase-LOVE can be applied to many applications such as short text retrieval, record linkage, synonym detection etc.

New Positional Encodings We reveal that exiting positional encodings have one important property, Symmetry, which has a potential weakness of insensitivity to global word swaps. One possible future work is to design a completely new positional encodings for addressing this issue.

Bibliography

[1] Mostafa Abdou, Vinit Ravishankar, Artur Kulmizev, and Anders Søgaard. Word order does matter and shuffled language models know it. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6907–6919, 2022.

[2] Eytan Adar. Sarad: A simple and robust abbreviation dictionary. *Bioinformatics*, 20(4):527–533, 2004.

[3] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. 2009.

[4] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9):137, 2018.

[5] Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 1995–2010. ACM, 2020.

[6] Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey. *arXiv preprint arXiv:1901.09069*, 2019.

[7] Abdulrahman Almuhareb. *Attributes in lexical acquisition*. PhD thesis, University of Essex, 2006.

[8] Hiroko Ao and Toshihisa Takagi. Alice: an algorithm to extract abbreviations from medline. *Journal of the American Medical Informatics Association*, 12(5):576–586, 2005.

[9] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[10] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[11] Edoardo Barba, Luigi Procopio, and Roberto Navigli. Extend: extractive entity disambiguation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2478–2488, 2022.

[12] Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, 2011.

[13] Suzanna Becker and Geoffrey E Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

[14] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*, 2018.

[15] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. 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)*, pages 3615–3620, 2019.

[16] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[17] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

[18] Luisa Bentivogli, Ido Kalman Dagan, Dang Hoa, Danilo Giampiccolo, and Bernardo Magnini. The fifth pascal recognizing textual entailment challenge. In *TAC 2009 Workshop*. no publisher, 2009.

[19] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[20] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[21] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[22] Rishi Bommasani, Kelly Davis, and Claire Cardie. Interpreting pretrained contextualized representations via reductions to static embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4758–4781, 2020.

[23] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

[24] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[25] Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480, 1992.

[26] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[27] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *Journal of artificial intelligence research*, 49:1–47, 2014.

[28] Razvan Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. 2006.

[29] Pere-Lluís Huguet Cabot and Roberto Navigli. Rebel: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, 2021.

[30] Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. Knowledgeable or educated guess? revisiting language models as knowledge bases. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1860–1874, 2021.

[31] Kris Cao and Marek Rei. A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 18–26, 2016.

[32] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1306–1313, 2010.

[33] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, 2017.

[34] Jeffrey T Chang, Hinrich Schütze, and Russ B Altman. Creating an online dictionary of abbreviations from medline. *Journal of the American Medical Informatics Association*, 9(6):612–620, 2002.

[35] Jean Charbonnier and Christian Wartena. Using word embeddings for unsupervised acronym disambiguation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2610–2619, 2018.

[36] Lihu Chen, Simon Razniewski, and Gerhard Weikum. Knowledge base completion for long-tail entities. *arXiv preprint arXiv:2306.17472*, 2023.

[37] Lihu Chen, Gael Varoquaux, and Fabian Suchanek. Imputing out-of-vocabulary embeddings with love makes languagemodels robust with little cost. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3488–3504, 2022.

[38] Lihu Chen, Gaël Varoquaux, and Fabian M Suchanek. A lightweight neural model for biomedical entity linking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12657–12665, 2021.

[39] Lihu Chen, Gael Varoquaux, and Fabian M Suchanek. Understanding the role of positional encodings in sentence representations. 2022.

[40] Lihu Chen, Gaël Varoquaux, and Fabian M Suchanek. Gladis: A general and large acronym disambiguation benchmark. In *The 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023)*, 2023.

[41] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.

[42] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[43] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. On sampling strategies for neural network-based collaborative filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 767–776, 2017.

[44] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. Knowledge graph completion: A review. *IEEE Access*, 8:192435–192456, 2020.

[45] Zheng Chen and Heng Ji. Collaborative ranking: A case study on entity linking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 771–781, 2011.

[46] Billy Chiu, Anna Korhonen, and Sampo Pyysalo. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st workshop on evaluating vector-space representations for NLP*, pages 1–6, 2016.

[47] Manuel Ciosici, Tobias Sommer, and Ira Assent. Unsupervised abbreviation disambiguation contextual disambiguation using word embeddings. *arXiv preprint arXiv:1904.00929*, 2019.

[48] Manuel R Ciosici and Ira Assent. Abbreviation expander-a web-based system for easy reading of technical documents. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 1–4, 2018.

[49] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[50] Louis Clouatre, Prasanna Parthasarathi, Amal Zouaq, and Sarath Chandar. Local structure matters most: Perturbation study in nlu. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3712–3731, 2022.

[51] World Wide Web Consortium et al. Rdf 1.1 concepts and abstract syntax. 2014.

[52] Maurizio Corbetta and Gordon L Shulman. Control of goal-directed and stimulus-driven attention in the brain. *Nature reviews neuroscience*, 3(3):201–215, 2002.

[53] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer, 2005.

[54] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.

[55] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

[56] Allan Peter Davis, Thomas C Wiegers, Michael C Rosenstein, and Carolyn J Mattingly. Medic: a practical disease vocabulary used at the comparative toxicogenomics database. *Database*, 2012, 2012.

[57] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *International Conference on Learning Representations*, 2020.

[58] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2018.

[59] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2019.

[60] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[61] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10, 2014.

[62] Rezarta Islamaj Dogan and Zhiyong Lu. An inference method for disease name normalization. In *2012 AAAI Fall Symposium Series*, 2012.

[63] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[64] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, Tim Finin, et al. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.

[65] Philipp Dufter, Martin Schmitt, and Hinrich Schütze. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763, 2022.

[66] William Edward Dyer. *Minimizing integration cost: A general theory of constituent order*. PhD thesis, University of California, Davis, 2017.

[67] Jennifer D'Souza and Vincent Ng. Sieve-based entity linking for the biomedical domain. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 297–302, 2015.

[68] Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, 2020.

[69] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard H. Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Trans. Assoc. Comput. Linguistics*, 9:1012–1031, 2021.

[70] István Endrédy and Attila Novák. More effective boilerplate removal-the goldminer algorithm. *Polibits*, (48):79–83, 2013.

[71] Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. Named entity disambiguation for noisy text. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 58–68, 2017.

[72] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.

[73] Zheng Fang, Yanan Cao, Qian Li, Dongjie Zhang, Zhenyu Zhang, and Yanbing Liu. Joint entity linking with deep reinforcement learning. In *The world wide web conference*, pages 438–447, 2019.

[74] Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*, 2016.

[75] Fernanda Ferreira, Karl GD Bailey, and Vittoria Ferraro. Good-enough representations in language comprehension. *Current directions in psychological science*, 11(1):11–15, 2002.

[76] Gregory P Finley, Serguei VS Pakhomov, Reed McEwan, and Genevieve B Melton. Towards comprehensive clinical abbreviation disambiguation using machine-labeled training data. In *AMIA Annual Symposium Proceedings*, volume 2016, page 560. American Medical Informatics Association, 2016.

[77] Matthew Francis-Landau, Greg Durrett, and Dan Klein. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1256–1261, 2016.

[78] Victoria Fromkin, Robert Rodman, and Nina Hyams. *An Introduction to Language (w/MLA9E Updates)*. Cengage Learning, 2018.

[79] Nobukazu Fukuda, Naoki Yoshinaga, and Masaru Kitsuregawa. Robust backed-off estimation of out-of-vocabulary embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4827–4838, 2020.

[80] Richard Futrell, Roger P Levy, and Edward Gibson. Dependency locality as an explanatory principle for word order. *Language*, 96(2):371–412, 2020.

[81] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[82] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[83] Abhishek Gattani, Digvijay S Lamba, Nikesh Garera, Mitul Tiwari, Xiaoyong Chai, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Entity extraction, linking, classification, and tagging for social media: a wikipedia-based approach. *Proceedings of the VLDB Endowment*, 6(11):1126–1137, 2013.

[84] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR, 2017.

[85] Omid Ghiasvand and Rohit J Kate. R.: Uwm: Disorder mention extraction from clinical text using crfs and normalization using learned edit distance patterns. In *In: Proc. SemEval 2014*. Citeseer, 2014.

[86] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9, 2007.

[87] Edward Gibson, Leon Bergen, and Steven T Piantadosi. Rational integration of noisy evidence and prior semantic expectations in sentence interpretation. *Proceedings of the National Academy of Sciences*, 110(20):8051–8056, 2013.

[88] John M Giorgi, Osvald Nitski, Gary D Bader, and Bo Wang. Declutr: Deep contrastive learning for unsupervised textual representations. *arXiv preprint arXiv:2006.03659*, 2020.

[89] Mauricio Girardi-Schappo, Ludmila Brochini, Ariadne A Costa, Tawan TA Carvalho, and Osame Kinouchi. Self-organized critical balanced networks: a unified framework. *arXiv preprint arXiv:1906.05624*, 2019.

[90] Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631, 2016.

[91] Joshua Goodman. Classes for fast maximum entropy training. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, volume 1, pages 561–564. IEEE, 2001.

[92] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[93] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[94] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274, 2009.

[95] Maosheng Guo, Yu Zhang, and Ting Liu. Gaussian transformer: a lightweight approach for natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6489–6496, 2019.

[96] Stephen Guo, Ming-Wei Chang, and Emre Kiciman. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1020–1030, 2013.

[97] Ashim Gupta, Giorgi Kvernadze, and Vivek Srikumar. Bert & family eat word salad: Experiments with text understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12946–12954, 2021.

[98] Nitish Gupta, Sameer Singh, and Dan Roth. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690, 2017.

[99] Prakhar Gupta and Martin Jaggi. Obtaining better static word embeddings using contextual embedding models. *arXiv preprint arXiv:2106.04302*, 2021.

[100] R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, 2006.

[101] Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414, 2012.

[102] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774, 2011.

[103] Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. *arXiv preprint arXiv:2203.16634*, 2022.

[104] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[105] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[106] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2020.

[107] Benjamin Heinzerling and Kentaro Inui. Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1772–1791, 2021.

[108] Benjamin Heinzerling and Michael Strube. Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. *arXiv preprint arXiv:1710.02187*, 2017.

[109] Aron Henriksson, Hans Moen, Maria Skeppstedt, Vidas Daudaravičius, and Martin Duneld. Synonym extraction and abbreviation expansion with ensembles of semantic spaces. *Journal of biomedical semantics*, 5(1):1–25, 2014.

[110] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.

[111] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[112] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[113] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 782–792, 2011.

[114] Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. Few-shot representation learning for out-of-vocabulary words. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4102–4112, 2019.

[115] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*, 2019.

[116] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[117] Rezarta Islamaj Dogan, G Craig Murray, Aurélie Névéol, and Zhiyong Lu. Understanding pubmed® user search behavior through log analysis. *Database*, 2009, 2009.

[118] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[119] Alpa Jain, Silviu Cucerzan, and Saliha Azzam. Acronym-expansion recognition and ranking on the web. In *2007 IEEE International Conference on Information Reuse and Integration*, pages 209–214. IEEE, 2007.

[120] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 1148–1158, 2011.

[121] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Networks Learn. Syst.*, 33(2):494–514, 2022.

[122] Zongcheng Ji, Qiang Wei, and Hua Xu. Bert-based ranking for biomedical entity normalization. *AMIA Summits on Translational Science Proceedings*, 2020:269, 2020.

[123] Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. X-factr: Multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5943–5959, 2020.

[124] Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.

[125] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[126] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

[127] Qiao Jin, Jinling Liu, and Xinghua Lu. Deep contextualized biomedical abbreviation expansion. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 88–96, 2019.

[128] Kun Jing and Jungang Xu. A survey on neural network language models. *arXiv preprint arXiv:1906.03591*, 2019.

[129] Zhao Jinman, Shawn Zhong, Xiaomin Zhang, and Yingyu Liang. Pbos: Probabilistic bag-of-subwords for generalizing word embedding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 596–611, 2020.

[130] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[131] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, 2017.

[132] Martin Josifoski, Nicola De Cao, Maxime Peyrard, Fabio Petroni, and Robert West. GenIE: Generative information extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4626–4643, July 2022.

[133] Ning Kang, Bharat Singh, Zubair Afzal, Erik M van Mulligen, and Jan A Kors. Using rule-based natural language processing to improve disease normalization in biomedical text. *Journal of the American Medical Informatics Association*, 20(5):876–881, 2013.

[134] Nora Kassner and Hinrich Schütze. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, 2020.

[135] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*, 2021.

[136] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, 2018.

[137] Yeachan Kim, Kang-Min Kim, Ji-Min Lee, and SangKeun Lee. Learning to generate word representations using subword information. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2551–2561, 2018.

[138] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[139] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[140] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[141] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[142] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European conference on machine learning*, pages 217–226. Springer, 2004.

[143] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of machine translation summit x: papers*, pages 79–86, 2005.

[144] Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. End-to-end neural entity linking. *CoNLL 2018*, page 519, 2018.

[145] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466, 2009.

[146] Cheng-Ju Kuo, Maurice HT Ling, Kuan-Ting Lin, and Chun-Nan Hsu. Bioadi: a machine learning approach to identifying abbreviations and definitions in biological literature. In *BMC bioinformatics*, volume 10, pages 1–10. BioMed Central, 2009.

[147] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[148] Leah S Larkey, Paul Ogilvie, M Andrew Price, and Brenden Tamilio. Acrophile: an automated acronym extractor and server. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 205–214, 2000.

[149] Phong Le and Ivan Titov. Distant learning for entity linking with automatic noise detection. In *57th Annual Meeting of the Association for Computational Linguistics*, pages 4081–4090. ACL Anthology, 2019.

[150] Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. Dnorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, 29(22):2909–2917, 2013.

[151] Robert Leaman and Zhiyong Lu. Taggerone: joint named entity recognition and normalization with semi-markov models. *Bioinformatics*, 32(18):2839–2846, 2016.

[152] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[153] Douglas B Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[154] Roger Levy. A noisy-channel model of human sentence comprehension under uncertain input. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pages 234–243, 2008.

[155] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[156] Bin Li, Fei Xia, Yixuan Weng, Xiusheng Huang, Bin Sun, and Shutao Li. Simclad: A simple framework for contrastive learning of acronym disambiguation. *arXiv preprint arXiv:2111.14306*, 2021.

[157] Bofang Li, Aleksandr Drozd, Tao Liu, and Xiaoyong Du. Subword-level composition functions for learning word embeddings. In *Proceedings of the second workshop on subword/character level models*, pages 38–48, 2018.

[158] Chao Li, Lei Ji, and Jun Yan. Acronym disambiguation using word embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[159] Haodi Li, Qingcai Chen, Buzhou Tang, Xiaolong Wang, Hua Xu, Baohua Wang, and Dong Huang. Cnn-based ranking for biomedical entity normalization. *BMC bioinformatics*, 18(11):79–86, 2017.

[160] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70, 2020.

[161] Yang Li, Bo Zhao, Ariel Fuxman, and Fangbo Tao. Guess me if you can: Acronym disambiguation for enterprises. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1308–1317, 2018.

[162] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4208–4215, 2018.

[163] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

[164] Gang Liu and Jiabao Guo. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, 2019.

[165] Haitao Liu, Chunshan Xu, and Junying Liang. Dependency distance: A new perspective on syntactic patterns in natural languages. *Physics of life reviews*, 21:171–193, 2017.

[166] Jie Liu, Caihua Liu, and Yalou Huang. Multi-granularity sequence labeling model for acronym expansion identification. *Information Sciences*, 378:462–474, 2017.

[167] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586, 2021.

[168] Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. Entity linking for tweets. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1304–1311, 2013.

[169] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Ro{bert}a: A robustly optimized {bert} pretraining approach, 2020.

[170] Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, 2019.

[171] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*, 2018.

[172] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[173] Minh-Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the seventeenth conference on computational natural language learning*, pages 104–113, 2013.

[174] Edward Ma. Nlp augmentation. https://github.com/makcedward/nlpaug, 2019.

[175] Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. Charbert: Character-aware pre-trained language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50, 2020.

[176] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, 2014.

[177] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[178] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

[179] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[180] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[181] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014.

[182] Sunil Mohan and Donghui Li. Medmentions: A large biomedical corpus annotated with umls concepts. In *Automated Knowledge Base Construction (AKBC)*, 2018.

[183] Francis Mollica, Matthew Siegelman, Evgeniia Diachek, Steven T Piantadosi, Zachary Mineroff, Richard Futrell, Hope Kean, Peng Qian, and Evelina Fedorenko. Composition is the core driver of the language-selective network. *Neurobiology of Language*, 1(1):104–134, 2020.

[184] Sean Monahan, John Lehmann, Timothy Nyberg, Jesse Plymale, and Arnold Jung. Cross-lingual cross-document coreference with entity linking. In *TAC*, 2011.

[185] Jose G Moreno, Romaric Besançon, Romain Beaumont, Eva D'hondt, Anne-Laure Ligozat, Sophie Rosset, Xavier Tannier, and Brigitte Grau. Combining word and entity embeddings for entity linking. In *The Semantic Web: 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28–June 1, 2017, Proceedings, Part I 14*, pages 337–352. Springer, 2017.

[186] Dana Movshovitz-Attias and William Cohen. Alignment-hmm-based extraction of abbreviations from biomedical text. In *BioNLP: Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, pages 47–55, 2012.

[187] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[188] David Nadeau and Peter D Turney. A supervised learning approach to acronym identification. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 319–329. Springer, 2005.

[189] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial intelligence*, 193:217–250, 2012.

[190] Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st workshop on vector space modeling for natural language processing*, pages 39–48, 2015.

[191] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[192] Naoaki Okazaki and Sophia Ananiadou. Building an abbreviation dictionary using a term recognition approach. *Bioinformatics*, 22(24):3089–3095, 2006.

[193] Yasumasa Onoe and Greg Durrett. Fine-grained entity typing for domain independent entity linking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8576–8583, 2020.

[194] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[195] Juri Opitz and Sebastian Burst. Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*, 2019.

[196] Serguei Pakhomov, Ted Pedersen, and Christopher G Chute. Abbreviation and acronym disambiguation in clinical discourse. In *AMIA Annual Symposium Proceedings*, volume 2005, page 589. American Medical Informatics Association, 2005.

[197] Chunguang Pan, Bingyan Song, Shengguang Wang, and Zhipeng Luo. Bert-based acronym disambiguation with multiple training strategies. *arXiv preprint arXiv:2103.00488*, 2021.

[198] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278, 2004.

[199] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*, 2005.

[200] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, 2016.

[201] Youngja Park and Roy J Byrd. Hybrid text mining for finding abbreviations and their definitions. In *Proceedings of the 2001 conference on empirical methods in natural language processing*, 2001.

[202] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[203] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. Yago 4: A reason-able knowledge base. In *European Semantic Web Conference*, pages 583–596. Springer, 2020.

[204] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[205] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

[206] Matthew E Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. Knowledge enhanced contextual word representations. 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)*, pages 43–54, 2019.

[207] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? 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)*, pages 2463–2473, 2019.

[208] Thang Pham, Trung Bui, Long Mai, and Anh Nguyen. Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1145–1160, 2021.

[209] Aleksandra Piktus, Necati Bora Edizel, Piotr Bojanowski, Édouard Grave, Rui Ferreira, and Fabrizio Silvestri. Misspelling oblivious word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 3226–3234, 2019.

[210] Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. Mimicking word embeddings using subword rnns. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, 2017.

[211] Sameer Pradhan, Noemie Elhadad, Brett R South, David Martinez, Lee M Christensen, Amy Vogel, Hanna Suominen, Wendy W Chapman, and Guergana K Savova. Task 1: Share/clef ehealth evaluation lab 2013. In *CLEF (Working Notes)*, pages 212–31, 2013.

[212] Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2021.

[213] Ofir Press, Noah A Smith, and Mike Lewis. Shortformer: Better language modeling using shorter inputs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5493–5505, 2021.

[214] Vera Provatorova, Samarth Bhargav, Svitlana Vakulenko, and Evangelos Kanoulas. Robustness evaluation of entity disambiguation using prior probes: the case of entity overshadowing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10501–10510, 2021.

[215] Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, 2019.

[216] James Pustejovsky, José Castano, Brent Cochran, Maciej Kotecki, and Michael Morrell. Automatic extraction of acronym-meaning pairs from medline databases. In *MEDINFO 2001*, pages 371–375. IOS Press, 2001.

[217] Guanghui Qin and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5203–5212. Association for Computational Linguistics, 2021.

[218] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[219] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[220] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[221] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[222] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[223] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, 2018.

[224] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[225] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 1375–1384, 2011.

[226] Simon Razniewski, Andrew Yates, Nora Kassner, and Gerhard Weikum. Language models as or for knowledge bases. *arXiv preprint arXiv:2110.04888*, 2021.

[227] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

[228] Stephen E Robertson, Steve Walker, Susan Jones, et al. Okapi at trec-3. 1995.

[229] Gaetano Rossiello, Md. Faisal Mahbub Chowdhury, Nandana Mihindukulasooriya, Owen Cornec, and Alfio Gliozzo. Knowgl: Knowledge generation and linking from text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[230] Erik Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[231] Shota Sasaki, Jun Suzuki, and Kentaro Inui. Subword-based compact reconstruction of word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3498–3508, 2019.

[232] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507, 2020.

[233] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2018.

[234] Timo Schick and Hinrich Schütze. Attentive mimicking: Better word embeddings by attending to informative contexts. *arXiv preprint arXiv:1904.01617*, 2019.

[235] Timo Schick and Hinrich Schütze. Learning semantic representations for novel words: Leveraging both form and context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6965–6973, 2019.

[236] Timo Schick and Hinrich Schütze. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8766–8774, 2020.

[237] Ariel S Schwartz and Marti A Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Biocomputing 2003*, pages 451–462. World Scientific, 2002.

[238] Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. Neural entity linking: A survey of models based on deep learning. *Semantic Web*, (Preprint):1–44, 2022.

[239] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[240] Wei Shen, Yuhan Li, Yinan Liu, Jiawei Han, Jianyong Wang, and Xiaojie Yuan. Entity linking meets deep learning: Techniques and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[241] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, 2014.

[242] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*, pages 449–458, 2012.

[243] Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. Incremental knowledge base construction using deepdive. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, volume 8, page 1310. NIH Public Access, 2015.

[244] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, 2020.

[245] Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. *Advances in neural information processing systems*, 32, 2019.

[246] Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. Neural cross-lingual entity linking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[247] Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2888–2913, 2021.

[248] Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. Overview of biocreative ii gene mention recognition. *Genome biology*, 9(2):1–19, 2008.

[249] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[250] Sunghwan Sohn, Donald C Comeau, Won Kim, and W John Wilbur. Abbreviation definition identification based on automatic precision estimates. *BMC bioinformatics*, 9(1):1–10, 2008.

[251] Robyn Speer, Catherine Havasi, et al. Representing general relational knowledge in conceptnet 5. In *LREC*, volume 2012, pages 3679–86, 2012.

[252] Mark Stevenson, Yikun Guo, Abdulaziz Alamri, and Robert Gaizauskas. Disambiguation of biomedical abbreviations. In *Proceedings of the BioNLP 2009 Workshop*, pages 71–79, 2009.

[253] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[254] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[255] Fabian M Suchanek, Jonathan Lajus, Armand Boschin, and Gerhard Weikum. Knowledge representation and rule mining in entity-centric knowledge bases. *Reasoning Web. Explainable Artificial Intelligence: 15th International Summer School 2019, Bolzano, Italy, September 20–24, 2019, Tutorial Lectures*, pages 110–152, 2019.

[256] Haitian Sun, Tania Bedrax-Weiss, and William Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. 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)*, pages 2380–2390, 2019.

[257] Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert. *arXiv preprint arXiv:2003.04985*, 2020.

[258] Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. Modeling mention, context and entity with neural networks for entity disambiguation. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[259] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha P. Talukdar, and Yiming Yang. A re-evaluation of knowledge graph completion methods. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5516–5522. Association for Computational Linguistics, 2020.

[260] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.

[261] Hanna Suominen, Sanna Salanterä, Sumithra Velupillai, Wendy W Chapman, Guergana Savova, Noemie Elhadad, Sameer Pradhan, Brett R South, Danielle L Mowery, Gareth JF Jones, et al. Overview of the share/clef ehealth evaluation lab

2013. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 212–231. Springer, 2013.

[262] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[263] Sho Takase and Naoaki Okazaki. Positional encoding to control output sequence length. *arXiv preprint arXiv:1904.07418*, 2019.

[264] David Temperley and Daniel Gildea. Minimizing syntactic dependency lengths: Typological/cognitive universal? *Annual Review of Linguistics*, 4:67–80, 2018.

[265] Jörg Tiedemann. Finding alternative translations in a large corpus of movie subtitle. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3518–3522, 2016.

[266] Matthew J Traxler. Trends in syntactic parsing: Anticipation, bayesian estimation, and good-enough parsing. *Trends in cognitive sciences*, 18(11):605–611, 2014.

[267] Asahi Ushio and Jose Camacho-Collados. T-ner: An all-round python library for transformer-based named entity recognition. *arXiv preprint arXiv:2209.12616*, 2022.

[268] Ahmet Üstün, Murathan Kurfalı, and Burcu Can. Characters or morphemes: How to represent words? Association for Computational Linguistics, 2018.

[269] Karin Vadovičová. Affective and cognitive prefrontal cortex projections to the lateral habenula in humans. *Frontiers in human neuroscience*, 8:819, 2014.

[270] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[271] Amir Pouran Ben Veyseh, Franck Dernoncourt, Walter Chang, and Thien Huu Nguyen. Maddog: A web-based system for acronym identification and disambiguation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 160–167, 2021.

[272] Amir Pouran Ben Veyseh, Franck Dernoncourt, Quan Hung Tran, and Thien Huu Nguyen. What does this acronym mean? introducing a new dataset for acronym identification and disambiguation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3285–3301, 2020.

[273] Amir Pouran Ben Veyseh, Nicole Meister, Seunghyun Yoon, Rajiv Jain, Franck Dernoncourt, and Thien Huu Nguyen. Macronym: A large-scale dataset for multilingual and multi-domain acronym extraction. *arXiv preprint arXiv:2202.09694*, 2022.

[274] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[275] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[276] Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. On position embeddings in bert. In *International Conference on Learning Representations*, 2020.

[277] Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. In *International Conference on Learning Representations*, 2019.

[278] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. Evaluating word embedding models: Methods and experimental results. *APSIPA transactions on signal and information processing*, 8, 2019.

[279] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[280] Shuohang WANG and Jing JIANG. A compare-aggregate model for matching text sequences.(2017). In *ICLR 2017: International Conference on Learning Representations, Toulon, France, April 24-26: Proceedings*, pages 1–15.

[281] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[282] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

[283] Yu-An Wang and Yun-Nung Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6840–6849, 2020.

[284] Gerhard Weikum, Xin Luna Dong, Simon Razniewski, Fabian Suchanek, et al. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Foundations and Trends® in Databases*, 10(2-4):108–490, 2021.

[285] Zhi Wen, Xing Han Lu, and Siva Reddy. Medal: Medical abbreviation disambiguation dataset for natural language understanding pretraining. *arXiv preprint arXiv:2012.13978*, 2020.

[286] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, 2016.

[287] Adina Williams, Nikita Nangia, and Samuel Bowman. 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, Volume 1 (Long Papers)*, pages 1112–1122, 2018.

[288] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[289] Dustin Wright. *NormCo: Deep disease normalization for biomedical knowledge base construction*. PhD thesis, UC San Diego, 2019.

[290] Junshuang Wu, Richong Zhang, Yongyi Mao, Hongyu Guo, Masoumeh Soflaei, and Jinpeng Huai. Dynamic graph convolutional networks for entity linking. In *Proceedings of The Web Conference 2020*, pages 1149–1159, 2020.

[291] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, 2020.

[292] Yonghui Wu, Joshua C Denny, S Trent Rosenbloom, Randolph A Miller, Dario A Giuse, Lulu Wang, Carmelo Blanquicett, Ergin Soysal, Jun Xu, and Hua Xu. A long journey to short abbreviations: developing an open-source framework for clinical abbreviation recognition and disambiguation (card). *Journal of the American Medical Informatics Association*, 24(e1):e79–e86, 2017.

[293] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[294] Yonghui Wu, Jun Xu, Yaoyun Zhang, and Hua Xu. Clinical abbreviation disambiguation using neural word embeddings. In *Proceedings of BioNLP 15*, pages 171–176, 2015.

[295] Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*, 2020.

[296] Jun Xu, Hee-Jin Lee, Zongcheng Ji, Jingqi Wang, Qiang Wei, and Hua Xu. Uth_ccb system for adverse drug reaction extraction from drug labels at tac-adr 2017. In *TAC*, 2017.

[297] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, 2018.

[298] Baosong Yang, Longyue Wang, Derek F Wong, Lidia S Chao, and Zhaopeng Tu. Assessing the ability of self-attention networks to learn word order. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3635–3644, 2019.

[299] Xiyuan Yang, Xiaotao Gu, Sheng Lin, Siliang Tang, Yueting Zhuang, Fei Wu, Zhigang Chen, Guoping Hu, and Xiang Ren. Learning dynamic context augmentation for global 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)*, pages 271–281, 2019.

[300] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[301] Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. Docred: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, 2019.

[302] Stuart Yeates, David Bainbridge, and Ian H Witten. Using compression to identify acronyms in text. In *Proceedings of the Conference on Data Compression*, page 582, 2000.

[303] Hong Yu, George Hripcsak, and Carol Friedman. Mapping abbreviations to full forms in biomedical articles. *Journal of the American Medical Informatics Association*, 9(3):262–272, 2002.

[304] Hong Yu, Won Kim, Vasileios Hatzivassiloglou, and W John Wilbur. Using medline as a knowledge source for disambiguating abbreviations and acronyms in

full-text biomedical journal articles. *Journal of biomedical informatics*, 40(2):150–159, 2007.

[305] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.

[306] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

[307] Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong Lu. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data*, 6(1):1–9, 2019.

[308] Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. Position-aware attention and supervised data improve slot filling. In *Conference on Empirical Methods in Natural Language Processing*, 2017.

[309] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, 2019.

[310] Jinman Zhao, Sidharth Mudgal, and Yingyu Liang. Generalizing word embeddings using bag of subwords. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 601–606, 2018.

[311] Zhicheng Zheng, Fangtao Li, Minlie Huang, and Xiaoyan Zhu. Learning to link entities with knowledge base. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 483–491, 2010.

[312] Qiwei Zhong, Guanxiong Zeng, Danqing Zhu, Yang Zhang, Wangli Lin, Ben Chen, and Jiayu Tang. Leveraging domain agnostic and specific knowledge for acronym disambiguation. *arXiv preprint arXiv:2107.00316*, 2021.

[313] Zexuan Zhong, Dan Friedman, and Danqi Chen. Factual probing is [MASK]: learning vs. learning to recall. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5017–5033. Association for Computational Linguistics, 2021.

[314] Danqing Zhu, Wangli Lin, Yang Zhang, Qiwei Zhong, Guanxiong Zeng, Weilin Wu, and Jiayu Tang. At-bert: Adversarial training bert for acronym identification winning solution for sdu@ aaai-21. *arXiv preprint arXiv:2101.03700*, 2021.

[315] Ganggao Zhu and Carlos A Iglesias. Exploiting semantic similarity for named entity disambiguation in knowledge graphs. *Expert Systems with Applications*, 101:8–24, 2018.

[316] Yi Zhu, Ivan Vulić, and Anna Korhonen. A systematic study of leveraging subword information for learning word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 912–932, 2019.

[317] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

[318] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. Robust and collective entity disambiguation through semantic embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 425–434, 2016.

Appendix for Chapter 6

A.1 Details of Experiments

A.1.1 Visualizations of Positional Encodings

Model	Size	Version	Language
BERT	110M	*bert-base-uncased*	English
DeBERTa	100M	*microsoft/deberta-base*	English
XLNet	110M	*xlnet-base-cased*	English

Table A.1: Details of pre-trained language models used in visualizations.

To understand what positional encodings learn after pre-training, we visualize the positional weights in attentional heads. The Identical Word Probing is adopted in this experiment [276]. The used pre-trained language models are shown in Table A.1, and the repeated words are randomly selected from the corresponding vocabulary. Note that sub-tokens like single characters and non-physical words are removed. For visualization, we adopt the Identical Word Probing proposed [276], which feeds many repeated identical words to pre-trained language models and thus the attention values ($\alpha_{i,j}$ in Equation 6.7) are disentangled with contextual weights. More specifically, we randomly select 100 words from the corresponding vocabulary (filtering out single characters and sub-words such as "`##nd`"). We repeat each word to compose a sentence of length 128. These 100 sentences are fed into a language model and the attention weights across different layers are averaged as the positional weight matrix of a particular language model.

A.1.2 Word Swap Probing

To valid if language models with positional encodings are sensitive to the local and global word swaps, we construct Shuffle-x and Shuffle-SR SNLI datasets. Shuffle-x

Model	Size	Version	Fine-tuned by us
BERT	110M	*bert-base-uncased*	✓
ALBERT	223M	*ynie/albert-xxlarge-v2-snli_mnli_fever_anli_R1_R2_R3nli*	×
DeBERTa	100M	*microsoft/deberta-base*	✓
XLNet	340M	*ynie/xlnet-large-cased-snli_mnli_fever_anli_R1_R2_R3-nli*	×
StrucBERT	340M	*bayartsogt/structbert-large*	✓

Table A.2: Details of pre-trained language models used in word swap probing.

means the word orders of phrases with length x are disrupted, e.g. "*an electric guitar*" is a 3-gram phrase, and it might be "*guitar an electric*" in Shuffle-3 SNLI. Through this way, a new sentence with the same meaning can be obtained and therefore the initial label of the sample will not be changed. To construct such shuffled datasets, the premise sentences in SNLI test set are shuffled and we keep the hypothesis sentences intact. Here, we let $x \in [3,6]$ and select a subset from SNLI to make sure that every premise sentence has at least one phrase with length from 2 to 6. We select five types of target phrases for shuffling: *Noun Phrase, Verb Phrase, Prepositional Phrase, Adverb Phrase, and Adjective Phrase*. Finally, a Shuffle-x SNLI is obtained by disrupting the order inside a phrase with length x and the size for each shuffle-x is around 5000. The first fourth rows in Table 6.2 shows some samples.

As for the Shuffle-SR SNLI dataset, the semantic roles of agent and patient are swapped in a sentence. We collect a subset from SNLI test set. This algorithm is applied successively to the premise and hypothesis sentence for a sample whose label is entailment, and if the result of either of them is not null, we consider it a valid shuffled sample, which means we only shuffle the premise or hypothesis. After, we can obtain a new sample and the pair of sentences are contradicted with each other. In total, there are 1329 samples. To ensure that all sentences are semantically correct, we manually selected 300 pairs from them. The last two rows in Table 6.2 shows two examples in Shuffle-SR dataset.

To probe the capabilities of language models on our newly constructed datasets, we adopt five different pre-trained language models (as shown in Table A.2) and we use Hugging Face for implementation [288]. These models are fine-tuned on the training set of SNLI, and the model with the best score on validation set is stored for the follow experiments. Note that there are off-the-shell ALBERT and XLNet for natural language inference, we therefore use them directly without fine-tuning. During fine-tuning stage, the maximum length of the tokenized input sentence pair is 128, and the optimizer is Adam [139] with learning rate of 2e-5. The batch size is 32 and the epoch is 3. After fine-tuning, the best model is evaluated on our shuffle SNLI test set, an we record their performances when faced with local and global word swaps.

A.1.3 Linguistic Discussions of Locality and Symmetry

Locality means that the positional weights favor the combination of units in a sentence to their adjacent units when creating higher-level representations. For example, sub-tokens can be composed into lexical meanings (e.g., {*"context","##ual"*} $\rightarrow$ "*contextual*") or words can be composed into phrase-level meaning (e.g., {*"take","off"*} $\rightarrow$ "*take off*"), and clause-level and sentence-level meaning can be obtained through an iterative combination of low-level meanings, which is consistent with the multi-layer structure in pre-trained language models. From a linguistic perspective, words linked in a syntactic dependency should be close in linear order, which forms what can be called a dependency locality [80]. Dependency locality provides a potential explanation for the formal features of natural language word order. Consider the two sentences *"John throws out the trash"* and *"John throws the trash out"*. Both are grammatically correct. There is a dependency relationship between *"throws"* and *"out"* and the verb is modified by the adverb. However, language users prefer the expression with the first sentence because it has a shorter total dependency length [66, 165, 264]. Based on the visualizations and dependency locality, we, therefore, speculate that one main function that positional encodings have learned during pre-training is local composition, which exists naturally in our understanding of sentences. Empirical studies also demonstrate that performances of shuffled language models are correlated with the violation of local structure [136, 50].

The *symmetry* (also observed by the two work [283, 276]) of the positional matrices implies that the contributions of forward and backward sequences are equal when combining adjacent units under the locality constraint. This is contrary to our intuition, as the forward and backward tokens play different roles in the grammar, as we have seen in the examples of "*a man playing an electric guitar on stage*" and "*an electric guitar playing a man on stage*". However, this symmetry is less disruptive at the local level inside sentences. Recent work in psycholinguistics has shown that sentence processing mechanisms are well designed for coping with word swaps [75, 154, 87, 266]. Further, this work [183] hypothesizes that the composition process is robust to local word violations. Consider the following example:

a. *on their last day they were overwhelmed by farewell messages and gifts*

b. *on their last day they were overwhelmed by farewell and messages gifts*

c. *on their last they day were overwhelmed farewell messages by and gifts*

The local word swaps (colored underlined words) are introduced in the latter two sentences, leading to a less syntactically well-formed structure. However, experimental results show that the neural response (fMRI blood oxygen level-dependent) in the language region does not decrease when dealing with word order degradation [183], suggesting that human sentence understanding is robust to local word swaps. Likewise, symmetry can be understood in this way: when a reader processes a word in a sentence, the forward and backward nearby words are the most combinable, and the comprehension of this composition is robust to its inside order. On the other hand,

symmetry is not an ideal property for sentence representations (consider the case of "*an electricity guitar*"), and we show the flaws of symmetry in the word swap probing task in Section 6.3.4.

Listing A.1 shows a code example about how to inject handcrafted positional encodings to the BERT backbone. Each variant is fine-tuned on the training dataset with different learning rates (among 9e-5, 7e-5, 5e-5, 3e-5, 1e-5). After, we evaluate the fine-tuned model on the Dev set and report the average score of five learning rates. Apart from BERT, we introduce the TUPE model as another baseline. Specifically, we pre-train the following variants:

- BERT is the original one and we use it as a baseline.
- BERT-A^* and BERT-I^* are varaints of the former two, but the encodings are learnable during pre-training.
- BERT-A^*-*s* shares learnable positional encodings within a layer.
- BERT-*only-c* is for ablation study, and the positional encodings $\mathbf{p}_i$ in Equation 6.2 are removed.
- BERT-*only-p* is for ablation study, and the contextual encodings $\mathbf{x}_i$ in Equation 6.2 are removed.
- BERT-A^*-*Seq* combines the two features in a sequential way, and the positional attentions are first used and then contextual attentions.
- ALiBi adds linear biases to contextual weights [212], and we apply a softmax layer to the original biases for obtaining a attention weight vector.
- ALiBi-*Seq* uses the same biases with ALiBi but combines the two features in a sequential way.

Suppose that the hidden dimension is 768, the layer number is 12, the head number is 12, and the maximum length is 512 for BERT_{base} model, we can calculate the size for each variant. The number of parameters of handcrafted positional encoding for each head is 262K (512×512). If positional heads are different across all layers, the total cost is 37.7M ($512 \times 512 \times 12 \times 12$). If the positional encodings are shared across heads, the total cost is 3.1M ($512 \times 512 \times 12$).

```
class MultiHeadedSelfAttention(nn.Module):
""" Multi-Headed Scaled Dot Product Attention """
def __init__(self, config):
super().__init__()
self.n_heads = config.n_heads
self.drop = nn.Dropout(config.p_drop_attn)
self.proj_q = nn.Linear(config.dim, config.dim)
self.proj_k = nn.Linear(config.dim, config.dim)
self.proj_v = nn.Linear(config.dim, config.dim)


def forward(self, x, mask, pe):
"""
x, q(query), k(key), v(value) : (B(batch_size), S(seq_len), D(dim))
mask : (B(batch_size) x S(seq_len))
pe: positional weights (B(batch_size), H(Head_number)), S(seq_len), S(seq_len))
* split D(dim) into (H(n_heads), W(width of head)) ; D = H * W
"""
# (B, S, D) -proj-> (B, S, D) -split-> (B, S, H, W) -trans-> (B, H, S, W)
q, k, v = self.proj_q(x), self.proj_k(x), self.proj_v(x)
q, k, v = (split_last(x, (self.n_heads, -1)).transpose(1, 2)
for x in [q, k, v])
# (B, H, S, W) @ (B, H, W, S) -> (B, H, S, S) -softmax-> (B, H, S, S)
scores = q @ k.transpose(-2, -1) / np.sqrt(k.size(-1))

# inject positional weights into contextual weights
# (B, H, S, S) + (B, H, S, S) -> (B, H, S, S)
scores = scores + pe

if mask is not None:
mask = mask[:, None, None, :].float()
scores -= 10000.0 * (1.0 - mask)

scores = self.drop(F.softmax(scores, dim=-1))
# (B, H, S, S) @ (B, H, S, W) -> (B, H, S, W) -trans-> (B, S, H, W)
h = (scores @ v).transpose(1, 2).contiguous()
# -merge-> (B, S, D)
h = merge_last(h, 2)
return h
```

Listing A.1: A code example of how to inject handcrafted positional encodings into self-attentions.

To inject handcrafted positional encodings, we pre-compute the positional weights and add them to the contextual weights directly, as shown in Listing A.1. These weights can be either frozen or learnable during pre-training. The code of the sequence combination is shown in Listing A.2.

```
class Sequence(nn.Module):
""" Sequence Block """

def __init__(self, config):
super().__init__()
self.pos_mode = config.pos_mode
self.pos_learnable = config.pos_learnable
self.self_attention = MultiHeadedSelfAttention(config)
self.positional_attention = PositionalAttention(config, learnable=self.pos_learnable)

def forward(self, x, mask):
# positional attention
pa = self.positional_attention(x, mask)
# contextual attention
sa = self.self_attention(pa, mask)
return sa
```

Listing A.2: A code example of the Sequence combination of positional and contextual features.

A.1.4 Details of Downstream Datasets

SentEval is based on a set of existing text classification tasks involving one or two sentences as input. However, most tasks in SentEval are closely related to sentiment

analysis and thus not diverse enough. GLUE benchmark introduces a series of difficult natural language understanding tasks while some particular tasks only contain one dataset, e.g., sentiment analysis and textual similarity. Moreover, the size of WNLI in GLUE is rather small and the GLUE webpage notes that there are issues with the construction of this dataset [1]. To better evaluate the capability of models for sentence representation, we, therefore, select 10 datasets from SentEval and GLUE, covering four types of sentence-level tasks:

- **Sentiment Analysis** is also known as opinion mining, which aims to classify the polarity of a given text, whether the expressed opinion is positive, negative, or neutral. We use MR [199], SUBJ [198], and SST [249] for this task.
- **Textual Entailment** describes the inference relation between a pair of sentences, whether the premise sentence entails the hypothesis sentence. Actually, this is a classification task with three labels: entailment, contradiction, and neutral. Here, we use QNLI [224], RTE [53, 100, 86, 18] and MNLI [287] for evaluation. Note that we report the average score for the two test sets of MNLI.
- **Paraphrase Identification** is to determine whether a pair of sentences have the same meaning. We use MRPC [63] and QQP [2] for evaluation.
- **Textual Similarity** deals with determining how similar two pieces of texts are. We use STS-B [33] and SICK-R [176] for evaluation.

```
class MultiHeadPositionalAttention(nn.Module):
""" Multi-Headed Scaled Dot Product Attention """
def __init__(self, config):
super().__init__()
self.n_heads = config.n_heads
self.drop = nn.Dropout(config.p_drop_attn)


def forward(self, x, mask, pe):
"""
x, q(query), k(key), v(value) : (B(batch_size), S(seq_len), D(dim))
mask : (B(batch_size) x S(seq_len))
pe: positional weights (B(batch_size), H(Head_number)), S(seq_len), S(seq_len))
* split D(dim) into (H(n_heads), W(width of head)) ; D = H * W
"""
# (B, S, D) -proj-> (B, S, D) -split-> (B, S, H, W) -trans-> (B, H, S, W)
q, k, v = (split_last(x, (self.n_heads, -1)).transpose(1, 2)
for x in [q, k, v])
# (B, H, S, W) @ (B, H, W, S) -> (B, H, S, S) -softmax-> (B, H, S, S)

scores = pe
if mask is not None:
scores.masked_fill_(~mask, 0.)

# (B, H, S, S) @ (B, H, S, W) -> (B, H, S, W) -trans-> (B, S, H, W)
h = (scores @ v).transpose(1, 2).contiguous()
# -merge-> (B, S, D)
h = merge_last(h, 2)
return h
```

Listing A.3: A code example of the Positional Attention.

1 https://gluebenchmark.com/faq

2 data.quora.com/First-Quora-Dataset-Release-Question-Pairs

A.2 Additional Experiments

A.2.1 Loss Curves of Pre-training

Apart from the performances on downstream tasks, the loss curves are also checked for different variants. For this goal, the training loss and validation loss are stored after certain steps. We use a hold-set as the validation set. As shown in Figure **??**, our proposed BERT-A* and BERT-A*-*Seq* have smaller loss than the original BERT. This can be observed again on the validation set.

A.2.2 Ablation Study of Positional and Contextual Encodings

To check the importance of positional and contextual Encodings, we conduct an ablation study. For this goal, the contextual encodings $\mathbf{x}_i$ or positional encodings $\mathbf{p}_i$ in Equation 6.2 are removed, respectively, during pre-training and the two new models are evaluated on 10 sentence-level datasets. As shown in Table A.3, the BERT-*only-c* and BERT-*only-p* both lag behind the original BERT models, which means the combination of the two features is beneficial for sentence representations. On the other hand, positional encodings are more important for sentiment analysis, and the cross-attentions from contextual embeddings matter in sentence-pair tasks.

Model	Sentiment Analysis			Textual Entailment			Paraphrase Identification		Textual Similarity		
	MR (22K)	SUBJ (20K)	SST-2 (68.8K)	QNLI (110K)	RTE (5.5K)	MNLI (413K)	MRPC (5.4K)	QQP (755k)	STS-B (8.4K)	SICK-R (9.4K)	Avg
BERT	$72.5_{\pm 5.3}$	$91.0_{\pm 2.7}$	$86.4_{\pm 2.7}$	$85.8_{\pm 1.0}$	$59.2_{\pm 1.2}$	$78.2_{\pm 0.8}$	$73.5_{\pm 1.8}$	$88.7_{\pm 0.6}$	$77.8_{\pm 4.1}$	$64.9_{\pm 6.0}$	77.8
BERT-*only-c*	$73.0_{\pm 4.6}$	$88.9_{\pm 2.5}$	$82.9_{\pm 0.4}$	$82.0_{\pm 0.1}$	$62.7_{\pm 4.3}$	$70.8_{\pm 0.8}$	$74.1_{\pm 0.3}$	$86.9_{\pm 0.5}$	$78.5_{\pm 0.3}$	$64.5_{\pm 5.7}$	76.2
BERT-*only-p*	$73.8_{\pm 4.6}$	$90.8_{\pm 1.4}$	$84.0_{\pm 0.7}$	$79.8_{\pm 1.2}$	$50.9_{\pm 1.4}$	$68.3_{\pm 1.0}$	$73.9_{\pm 1.6}$	$85.8_{\pm 0.6}$	$47.1_{\pm 18.0}$	$51.7_{\pm 9.2}$	70.6

Table A.3: Ablation study across 10 sentence-level tasks. We report the average score of five runs using different learning rates.

ECOLE
DOCTORALE

Titre: Vers des systèmes de ésambiguïsation d'entités efficaces, généraux et robustes

Mots clés: résolution d'entité, apprentissage automatique, traitement automatique du langage

Résumé: La désambiguïsation des entités vise à faire correspondre les mentions dans les documents à des entités standard dans une base de connaissances donnée, ce qui est important pour diverses applications telles que l'extraction d'informations, la recherche sur le web et la réponse aux questions. Bien que le domaine soit très dynamique et que de nombreux travaux nouveaux apparaissent, trois questions sont sous-explorées par les travaux antérieurs. 1) Peut-on utiliser un petit modèle pour approcher les performances d'un grand modèle ? 2) Comment développer un système de désambiguïsation unique adapté à plusieurs domaines ? 3) Les systèmes existants sont-ils robustes aux mots hors-vocabulaire et aux différents ordres de mots ? Sur la base de ces trois questions, nous étudions comment construire un système de désambiguïsation d'entités efficace, général et robuste. Nous appliquons également avec succès la désambiguïsation d'entités à la tâche d'achèvement de la base de connaissances, en particulier pour les entités à longue traîne.

Title: Towards Efficient, General, and Robust Entity Disambiguation Systems

Keywords: entity disambiguation, automatic language processing, natural language processing

Abstract: Entity disambiguation aims to map mentions in documents to standard entities in a given knowledge base, which is important for various applications such as information extraction, Web search and question answering. Although the field is very vibrant with many novel works popping up, there are three questions that are underexplored by prior work. 1) Can we use a small model to approach the performance of a big model? 2) How to develop a single disambiguation system adapted to multiple domains? 3) Are existing systems robust to out-of-vocabulary words and different word orderings? Based on the three questions, we explore how to construct an efficient, general and robust entity disambiguation system. We also successfully apply entity disambiguation to the knowledge base completion task, especially for the long-tail entities.

Institut Polytechnique de Paris
91120 Palaiseau, France