Representing Relational Knowledge with Language Models

Thesis submitted in accordance with the requirements of Cardiff University for the degree of Doctor in Philosophy by

Asahi Ushio

March 2024

Abstract

Relational knowledge is the ability to recognize the relationship between instances, and it has an important role in human understanding a concept or commonsense reasoning. We, humans, structure our knowledge by understanding individual instances together with the relationship among them, which enables us to further expand the knowledge. Nevertheless, modelling relational knowledge with computational models is a long-standing challenge in Natural Language Processing (NLP). The main difficulty at acquiring relational knowledge arises from the generalization capability.

For pre-trained Language Model (LM), in spite of the huge impact made in NLP, relational knowledge remains understudied. In fact, GPT-3 (Brown et al., 2020), one of the largest LM at the time being with 175 billions of parameters, has shown to perform worse than a traditional statistical baseline in an analogy benchmark. Our initial results hinted at the type of relational knowledge encoded in some of the LMs. However, we found out that such knowledge can be hardly extracted with a carefully designed method tuned on a task specific validation set.

According to such finding, we proposed a method (*RelBERT*) for distilling relational knowledge via LM fine-tuning. This method successfully retrieves flexible relation embeddings that achieve State-of-The-Art (SoTA) in various analogy benchmarks. Moreover, it exhibits a high generalization ability to be able to handle relation types that are not included in the training data. Finally, we propose a new task of modelling graded relation in named entities, which reveals some limitations of recent SoTA LMs as well as RelBERT, suggesting future research direction to model relational knowledge in the current LM era, especially when it comes to named entities.

Acknowledgements

This thesis compiles all the research achieved during the three year research activity at Cardiff University, one of the most challenging yet delightful and memorable time in my life.

I would like to thank everyone who help my study, as the thesis would not have been accomplished without their help. I begin to express the gratitude to my family, especially my wife, who decided to move all the way from Japan to U.K. together with me to support my study. I would like to thank my supervisors, Prof. Jose Camacho-Collados and Prof. Steven Schockaert, who always give insightful feedback and navigate the study to the right direction, in addition to the generous supports in every aspect for my PhD study.

I would like to thank Dr. Fernando Alva-Manchego, Dr. Luis Espinosa-Anke, Dr. Federico Liberatore, and Dr. Yi Zhou from the Natural Language Processing Group at Cardiff University (CardiffNLP), for all the useful discussions and support my research and get the papers accepted by top-tier conferences successfully. I would

also like to thank my collaborators Prof. Danushka Bollegala, Dr. Francesco Barbieri, Leonardo Neves, Vitor Silva Sousa, Dimosthenis Antypas, Joanne Boisson, Kiamehr Rezaee, Zara Siddique, Hsuvas Borkakoty, Dr. Daniel Loureiro, and Dr. Mark Anderson.

Lastly, I owe a debt of gratitude to the School of Computer Science & Informatics, Cardiff University for providing the scholarship to financially support my study.

Contents

List of Figures

List of Tables

Acronyms

NLP	Natural Language Processing
LRA	Latent Relational Analysis
LM	Language Model
MLM	Masked Language Model
CLM	Causal Language Model
AI	Artificial Intelligence
PMI	Point-wise Mutual Information
SGNS	Skip-Gram with Negative Sampling
CBOW	Continuous Bag of Words
LDA	Latent Dirichlet Allocation
LSTM	Long Short-Term Memory network

RNN	Recurrent Neural Network
NLG	Natural Language Generation
EDLM	Encoder-Decoder Language Model
CNN	Convolutional Neural Network
SoTA	State-of-The-Art
BPE	Byte Pair Encoding
mPPL	Marginal Likelihood biased Perplexity
AP	Analogical Proportion
InfoNCE	Information Noise Constrastive Estimation
InfoLOOB	Info-Leave-One-Out-Bound
SVD	Singular Value Decomposition
LSA	Latent Semantic Analysis
RelSim	Relational Similarity Dataset
QA	Question Answering
KG	Knowledge Graph
tSNE	t-Distributed Stochastic Neighbor Embedding
HDBSCAN	hierarchical Density-Based Spatial Clustering of Applications with Noise
LC	List Completion

NER	Named Entity Recognition
NLI	Natural Language Inferrence
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
MRR	Mean Reciprocal Rank
IR	Information Retrieval
P@5	Precision at 5
TPR	Topical PageRank
QG	Question Generation
AE	Answer Extraction
QAG	Question and Answer Generation

1 Introduction

Computational models to recognize the lexical relationship between two words have long been studied as a fundamental task in NLP (Turney, 2005). For instance, as a representative early work, Lin and Pantel (2001) relied on sentences in which the two target words co-occur (e.g. *London* and *U.K.*), using dependency paths to model the relationship between the word pair. Along similar lines, Turney (2005) relied on templates expressing lexical patterns to characterise word pairs (e.g. *[head word] is the capital of [tail word]*), thus again relying on sentences where the words co-occur. Since the advent of word embeddings, most approaches for modelling relations have relied on word vectors in one way or another. A common strategy to model the relation between two words was to take the vector difference between the embeddings of each word Mikolov et al. (2013a); Gladkova et al. (2016); Vylomova et al. (2016). For example, the relationship between "King" and "Queen" is the gender difference, which can be captured by $\mathsf{wv}(\text{King}) - \mathsf{wv}(\text{Queen})$, where wv is a word embedding model that returns the embedding of a word. Although the vector difference of word embeddings quickly gained popularity, it has been shown that the latent space of such relation vectors is noisy, with nearest neighbours often

Query:		wing:air
Candidates:	(1)	arm:hand
	(2)	lung:breath
	(3)	**flipper:water**
	(4)	cloud:sky
	(5)	engine:jet

Query:		perceptive:discern
Candidates:	(1)	determined:hesitate
	(2)	authoritarian:heed
	(3)	abandoned:neglect
	(4)	restrained:rebel
	(5)	**persistent:persevere**

Table 1.1: Two examples of analogy task from the SAT dataset, where the candidate in bold characters is the answer in each case.

corresponding to different relationships (Linzen, 2016; Drozd et al., 2016; Bouraoui et al., 2018). As another approach of modelling relationship as distributed representation, one can directly learn to model the relationship between a word pair on the sentences mentioning both words (Jameel et al., 2018a; Espinosa-Anke and Schockaert, 2018; Washio and Kato, 2018a; Camacho-Collados et al., 2019; Joshi et al., 2019; Washio and Kato, 2018b). These models can directly output a relation vector of a word pair, unlike the vector difference that is a byproduct of word embedding models, or lexical patterns or dependency paths to characterise relationships based on such sentences. However, relying solely on sentences containing both words assumes such sentences to always describe the relationship, although this does not hold true in practice, that results in noisy representations.

The study of relation understanding is active in Knowledge Graph (KG) such as Wikidata (Vrandečić and Krötzsch, 2014) or ConceptNet (Speer et al., 2017) as well. As an opposite to the distributed representation, KGs rely on symbolic representation of triples, that relies on fixed relational schema between word pairs explicitly. KGs usually suffer from its incompleteness due to the lack of word pairs or relation types. Moreover, KGs can not capture a fine-grained relation difference beyond the fixed relational schema, which is essential to solve relational knowledge-intensive task in practice. For example, Table 1.1 shows two instances of analogy

task from SAT (Turney, 2005), where the relationships found in the query are abstract. To solve it with a KG, we can expect a KG to contain triples such as (*wing*, *UsedFor*, *air*), but such knowledge is not sufficient to solve the given question, e.g. since all of (*lung*, *UsedFor*, *breath*), (*engine*, *UsedFor*, *jet*), and (*flipper*, *UsedFor*, *water*) make sense. The issue here is the relationship *UsedFor* is too vague and does not describe the relationship between *wing* and *air* accurately enough to solve the task. Such limitations of KGs are recently tackled with pre-trained LM, which are shown to remember factual knowledge implicitly from the textual corpus used to pre-train LMs (Petroni et al., 2019a; Jiang et al., 2020; Heinzerling and Inui, 2021). The core idea is to extract KG triples from LMs (West et al., 2022; Hao et al., 2022; Cohen et al., 2023) to augment a KG to scale it automatically with more coverage of broader triples. Such methods can help to alleviate the incompleteness of KGs, but the resulting representations are still too coarse-grained for many applications. Large LMs are also inefficient and difficult to control as it is not well-understood where the internal representation of such symbolic knowledge is stored in LMs (Kassner and Schütze, 2020).

In order to tackle the limitations in the aforementioned studies for modelling lexical relationship, we explore methodologies of leveraging pre-trained LMs (Devlin et al., 2019; Radford et al.) to recognize the lexical relationship in this thesis. Subsequently, we propose a lexical relation embedding model based on LM fine-tuning to leverage the relational knowledge stored in the LM. The rest of this introductory chapter is organised as follows: § 1.1 outlines the research aim and motivations for this thesis, and § 1.2 outlines the contribution of research, § 1.3 lists publications that are peer-reviewed or currently under review and § 1.4 describes the structure of the thesis and summarises this introductory chapter.

1.1 Motivation

Relations play a central role in many applications. For instance, many question answering models currently rely on ConceptNet for modelling the relation between the concepts that are mentioned in the question and a given candidate answer (Yasunaga et al., 2021; Sun et al., 2022; Jiang et al., 2022). Commonsense KGs are similarly used to provide additional context to computer vision systems, e.g. for generating scene graphs (Gu et al., 2019; Chen et al., 2023) and for visual question answering (Wu et al., 2022). Many recommendation systems also rely on KGs to identify and explain relevant items (Wang et al., 2018, 2019b). Other applications that rely on KGs, or on modelling relationships more broadly, include semantic search (Arguello Casteleiro et al., 2020; Jafarzadeh et al., 2022), flexible querying of relational databases (Bordawekar and Shmueli, 2017), schema matching (Fernandez et al., 2018), completion and retrieval of Web tables (Zhang et al., 2019a) and ontology completion (Bouraoui and Schockaert, 2019).

Many of the aforementioned applications rely on KGs, which are incomplete and limited in expressiveness due to their use of a fixed relation schema. Relation embeddings have the potential to address these limitations, especially in contexts which involve ranking or measuring similarity, where extracting knowledge by prompting large LMs cannot replace vector based representations. Relation embeddings can also provide a foundation for systems that rely on analogical reasoning, where we need to identify correspondences between a given scenario and previously encountered ones (Gentner and Markman, 1997).

Finally, by extracting relation embeddings from LMs, we can get more insight into

what knowledge is captured by such models, since these embeddings capture the knowledge from the model in a more direct way than what is possible with prompting based methods (Petroni et al., 2019b; Jiang et al., 2020). The common prediction-based model probing techniques (Petroni et al., 2019b; Jiang et al., 2020) can easily be controlled by adversarial input such as negation (Kassner and Schütze, 2020), and recent studies focus on finding a set of parameter association that represents factual knowledge about named entities (Meng et al., 2022). From this perspective, relation embedding can be seen as an explicit representation of the relational knowledge, and more reliable than prediction-based model probing on the relational knowledge (Bouraoui et al., 2020).

1.2 Contributions

This thesis aim to achieve the goal of understanding relational knowledge in LMs and developing a method to extract such relational knowledge from LM to apply in the downstream tasks with relation understanding. To this end, the thesis makes a number of noticeable contributions as listed below:

1. **Unification of Analogy Question Benchmarks and Empirical Evaluation.** The capability for modelling analogy of LMs can be seen as a proxy of relational knowledge of the LMs. The task of recognizing analogies, however, has received relatively little attention, in spite of the central role of analogy in human cognition. We address the lack of the LM evaluation on analogy modelling and have created a set of analogy questions from various domain including educational settings, commonsense KG, named entities, and scien-

tific metaphor. Moreover, we have shown the results of simple zero-shot approaches to solve the analogy questions with static word embeddings and LMs to establish the baselines. This work is a part of contribution from the paper that has been published as a long paper at the 59th Annual Meeting of the Association of Computational Linguistics (ACL 2021) and is further discussed in Chapter 3.

2. **Comprehensive Study of Language Model to Solve Analogy Question.** We have presented an extensive analysis of the ability of LMs to identify analogies. Our empirical results shed light on the strengths and limitations of various models. Our conclusion is that language models can identify analogies to a certain extent, but not all LMs are able to achieve a meaningful improvement over word embeddings (whose limitations in analogy tasks are well documented). On the other hand, when carefully tuned, some LMs are able to achieve SoTA results. We emphasize that results are highly sensitive to the chosen hyperparameters (which define the scoring function and the prompt among others). This work has been published as a long paper at the 59th Annual Meeting of the Association of Computational Linguistics (ACL 2021) and is further discussed in Chapter 3.

3. **Relation Embedding Model via Language Models Fine-tuning.** We have proposed a strategy for learning relation embeddings, *RelBERT*, i.e. vector representations of pairs of words which capture their relationship. The main idea is to fine-tune a pre-trained language model using the relational similarity dataset from SemEval 2012 Task 2, which covers a broad range of semantic relations. In our experimental results, we found the resulting relation embeddings to be of high quality, outperforming state-of-the-art methods on all the analogy questions and some of the relation classification bench-

marks. Crucially, we found that RelBERT is capable of modelling relationships that go well beyond those that are covered by the training data, including morphological relations and relations between named entities. Being based on RoBERTa$_{\text{LARGE}}$, our main RelBERT model has 354M parameters. This relatively small size makes RelBERT convenient and efficient to use in practice. Surprisingly, we found RelBERT to significantly outperform language models which are several orders of magnitude larger. This work has been published as a long paper at the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021) and is further discussed in Chapter 4.

4. **New Dataset of Graded Relation in Named Entities.** We have proposed the task of modelling graded relations between named entities, with a new dataset. The task consists in ranking entity pairs according to how much they satisfy a given graded relation, where models only have access to the description of the relation and five prototypical instances per relation. To assess the difficulty of the task, we analysed a large number of baselines, including public large LMs of up to 30B parameters, SoTA relation embedding models, and closed large LMs such as GPT-4. We found significant performance differences between the largest LMs and their smaller siblings, which highlights the progress achieved in NLP in the last few years by scaling up LMs. However, even the largest models trail human performance by more than 20 percentage points. This work is further discussed in Chapter 5.

5. **Open-source Libraries** We have developed four open-source software to facilitate the use of NLP technologies for any levels of practitioners. The list of libraries include keyword extraction, Named Entity Recognition (NER), NLP tools for social media, and Question Generation (QG), where all the works are

available via standard package distributors such as `pip`[1] and `GitHub`[2]. This work is further discussed in Chapter 7.

1.3 Publications

The majority of the material in this thesis have been published in peer-reviewed conferences in NLP, while the remaining material are under reviewed at peer-reviewed conferences.

1.3.1 Core Publications

Below is a list of the core publications where the main contributions of the thesis have been published in the chronological order.

- Asahi Ushio, Luis Espinosa Anke, Steven Schockaert, and Jose Camacho-Collados. 2021. BERT is to NLP what AlexNet is to CV: Can Pre-Trained Language Models Identify Analogies?. 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 3609–3624, Online. Association for Computational Linguistics. Chapter 3
- Asahi Ushio, Jose Camacho-Collados, and Steven Schockaert. 2021. Distilling Relation Embeddings from Pretrained Language Models. In Proceedings of

[1] https://pypi.org
[2] https://github.com/

the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9044–9062, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. Chapter 4

- Asahi Ushio, Jose Camacho-Collados, and Steven Schockaert. 2023. RelBERT: Modelling Relational Similarity with Language Models. *under review at The journal of Artificial Intelligence (AIJ)*. Chapter 4

- Asahi Ushio, Jose Camacho-Collados, and Steven Schockaert. 2023. A RelEntLess Benchmark
for Modelling Graded Relations between Named Entities. *under review at EMNLP 2023*. Chapter 5

1.3.2 Remaining Publications

Below is a list of the publications where the remaining contributions in the thesis have been published in the chronological order.

- Asahi Ushio and Jose Camacho-Collados. 2021. T-NER: An All-Round Python Library for Transformer-based Named Entity Recognition. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, pages 53–62, Online. Association for Computational Linguistics. Chapter 7

- Asahi Ushio, Federico Liberatore, and Jose Camacho-Collados. 2021. Back to the Basics: A Quantitative Analysis of Statistical and Graph-Based Term Weighting Schemes for Keyword Extraction. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 8089–8103,

Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. Chapter 7

- Asahi Ushio, Dimosthenis Antypas, Jose Camacho-Collados, Vitor Silva, Leonardo Neves, and Francesco Barbieri. 2022. Twitter Topic Classification. In Proceedings of the 29th International Conference on Computational Linguistics, pages 3386–3400, Gyeongju, Republic of Korea. International Committee on Computational Linguistics. Chapter 7

- Asahi Ushio, Francesco Barbieri, Vitor Sousa, Leonardo Neves, and Jose Camacho-Collados. 2022. Named Entity Recognition in Twitter: A Dataset and Analysis on Short-Term Temporal Shifts. In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 309–319, Online only. Association for Computational Linguistics. Chapter 7

- Jose Camacho-collados, Kiamehr Rezaee, Talayeh Riahi, Asahi Ushio, Daniel Loureiro, Dimosthenis Antypas, Joanne Boisson, Luis Espinosa Anke, Fangyu Liu, and Eugenio Martínez Cámara. 2022. TweetNLP: Cutting-Edge Natural Language Processing for Social Media. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–49, Abu Dhabi, UAE. Association for Computational Linguistics. Chapter 7

- Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. 2022. Generative Language Models for Paragraph-Level Question Generation. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 670–688, Abu Dhabi, United Arab Emirates. Associ-

ation for Computational Linguistics. Chapter 7

- Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. 2023. An Empirical Comparison of LM-based Question and Answer Generation Methods. In Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics, Toronto, Canada. Association for Computational Linguistics: Findings. Chapter 7
- Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. 2023. A Practical Toolkit for Multilingual Question and Answer Generation. In Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics, Toronto, Canada. Association for Computational Linguistics: System Demonstrations. Chapter 7

1.4 Thesis Outline and Summary

In this section, we provide an overview of this thesis with a brief description to each chapter, including research aims, motivations, questions, and contributions. Figure 1.1 shows the entire structure of this thesis, where chapters with same interest are grouped together. The first set of chapters are preliminary chapters to review the literature of relational representation learning and LMs (Chapter 1 and Chapter 2). The second set of chapters focus on the representation learning of the lexical relation. We present our study to understand relational knowledge in pretrained LMs, where we solve analogy questions with LMs in a various scoring function without fine-tuning, and discuss the relational knowledge of LMs with the performance on the analogy questions (Chapter 3). Then, we propose a framework of LM fine-tuning to distill the relational knowledge of LMs as relation embed-

ding, and we show the resulting relation embedding models are not only capable of achieving SoTA in many analogy questions, but shown to possess a high generalization ability to unseen relation types (Chapter 4). At last, we extend our study to more challenging task that is a graded relation modelling in between named entities by creating new datasets with human annotation (Chapter 5).

While we were studying for relational knowledge understanding toward completing the research described in this thesis, we have developed four major NLP open source software including keyword extraction § 7.1, NER § 7.2, NLP on social media § 7.3, and QG § 7.4, which are not directly related to the main topic of this thesis, yet are clearly contributions to the NLP community (Chapter 7). Keyword extraction § 7.1 often constructs a graph of keyword from a paragraph based on the co-occurrence, and the relationship between each keyword is a key factor to improve the quality of the keyword extraction. NER § 7.2 is a task to extract and classify the type of named entities from a sentence, where the named entity is related to our benchmark of relation between named entities we propose at the last in the thesis. The study of NER has been expanded to social media domain that ended up with a software to facilitate NLP application on social media § 7.3. Finally, QG § 7.4 is a task to generate question given an answer, which is often a named entity, and it requires to understand how the named entity is used in the paragraph.

In the next chapter, Chapter 2, we will explain the relevant literature on the relational knowledge understanding, as well as the basic of the computational models including static word embeddings, and LMs.

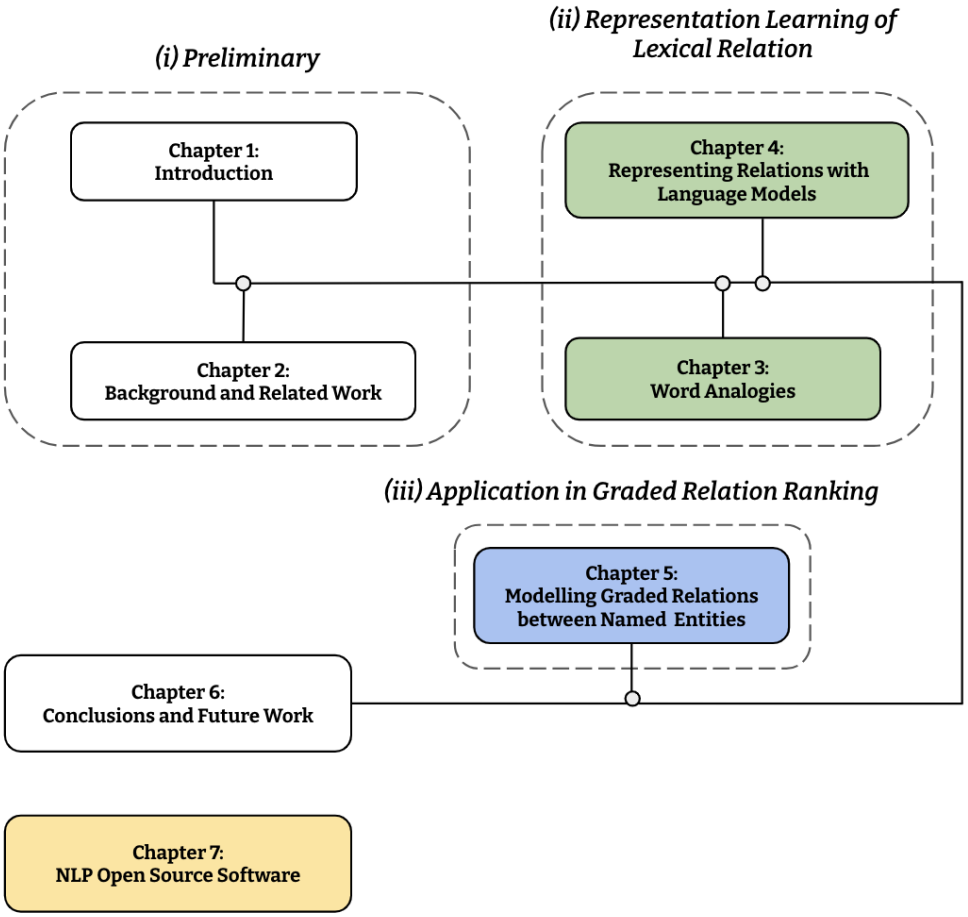

Figure 1.1: Outline of the thesis. The green box is the chapter investigating LMs for analogy modelling, blue boxes are the chapters studying relational representation learning with LMs, and yellow box is an independent chapter to introduce the NLP open source software we developed while studying for this thesis.

2

Background and Related Work

The representation of relationship has been studied in the context of the vector representations of words, known as word embeddings. The idea to obtain high-quality relation representation is motivated by the unsupervised relation discovery from documents. Since the advent of pre-trained LM, there had been an extensive study to uncover the internal mechanism, especially the knowledge stored in the LMs, which is often referred as "Knowledge Probing" of LMs. Despite of a number of LM probing research, relational knowledge of LMs is less studied. One of the common approach for understanding relational knowledge is to regard the ability of modelling analogy as a proxy of relational knowledge. Bearing this in mind, in this chapter, we review the literature of word embedding (§ 2.1), unsupervised relation discovery (§ 2.2), LMs (§ 2.3), general knowledge probing of LM (§ 2.4), understanding and application of relational knowledge in LM (§ 2.5), and modelling analogy (§ 2.6).

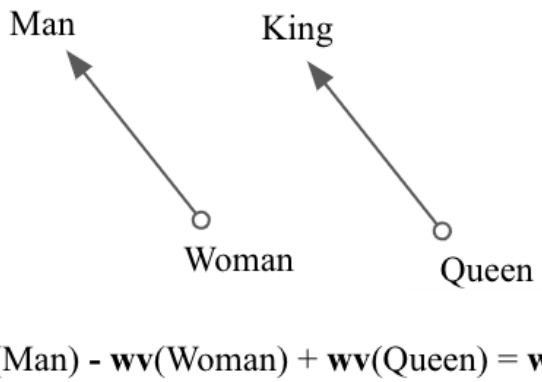

Figure 2.1: An illustration of wv(King) achieved as an addition of wv(King) and a relation embedding representing the gender difference that is wv(Man) − wv(Woman).

2.1 Word Embedding

Word embeddings (Mikolov et al., 2013a; Pennington et al., 2014; Bojanowski et al., 2016) are models that represent distributed semantic meaning of a word in a form of a fixed-length vector. Those word embeddings can be used as feature vectors to solve various NLP tasks (Zou et al., 2013; Bordes et al., 2014; Weiss et al., 2015), which have been shown to be more effective than hand-crafted features. Not only as a feature vector to supervised learning, but the word embeddings themselves are capable of representing the relationship in between words by the vector difference of their word embeddings, which means the relation between two words A and B is to some extent captured by the vector difference of their embeddings. Writing $\mathsf{wv}(A)$ for the word embedding of a word A, we can thus present the relation embedding of the form $\mathbf{x}_{(A,B)} = \mathsf{wv}(B) - \mathsf{wv}(A)$, as illustrated in Figure 2.1.

Word2vec (Mikolov et al., 2013a) is the most prevailing word embedding model,

where it has two variants of Continuous Bag of Words (CBOW) and Skip-Gram with Negative Sampling (SGNS). Both of the two models are a simple single-layer feed forward neural network with log-linear models, but they differ in the training objectives. CBOW predicts the target word from its surrounding words, while SGNS predicts the surrounding words using the target word.

GloVe (Pennington et al., 2014) is a word embedding model that takes the word co-occurrence matrix into account at learning embedding vector. It is a log-bilinear regression model that combines both of global matrix factorisation and local context window methods to produce word embeddings. One of the common issues for word embedding models is that they cannot handle unseen words out of their vocabulary. Bojanowski et al. (2016) proposed fastText, which is aimed at solving the out-of-vocabulary issue by deriving representations for unseen words from learnt n-gram embeddings. The idea stands on the hypothesis that languages should heavily rely on morphology and composition at constructing words, which can help to generalise to unseen words.

2.2 Unsupervised Relation Discovery

Modelling how different words are related is a long-standing challenge in NLP. An early approach is DIRT (Lin and Pantel, 2001), which encodes the relation between two nouns as the dependency path connecting them. Their view is that two such dependency paths are similar if the sets of word pairs with which they co-occur are similar. Along similar lines, (Hasegawa et al., 2004) cluster named entity pairs based on the bag-of-words representations of the contexts in which they appear. In (Yao et al., 2011), a generative probabilistic model inspired by Latent Dirichlet

Allocation (LDA) (Blei et al., 2003) was proposed, in which relations are viewed as latent variables (similar to topics in LDA). Turney (2005) proposed a method called Latent Relational Analysis (LRA), which uses matrix factorization to learn relation embeddings based on co-occurrences of word pairs and dependency paths. Matrix factorization is also used in the Universal Schema approach from Riedel et al. (2013), which represents entity pairs by jointly modelling (i) the contexts of occurrences of entity pairs in a corpus and (ii) the relational facts that are asserted about these entities in a given knowledge base. After the introduction of Word2Vec, several approaches were proposed that relied on word embeddings for summarising the contexts in which two words co-occur. For instance, (Jameel et al., 2018b) introduced a variant of the GloVe word embedding model, in which relation vectors are jointly learned with word vectors. In SeVeN (Espinosa-Anke and Schockaert, 2018) and RELATIVE (Camacho-Collados et al., 2019), relation vectors are computed by averaging the embeddings of context words, while pair2vec (Joshi et al., 2019) uses an Long Short-Term Memory network (LSTM) to summarise the contexts in which two given words occur, and (Washio and Kato, 2018a) learns embeddings of dependency paths to encode word pairs. Another line of work is based on the idea that relation embeddings should facilitate link prediction, i.e. given the first word and a relation vector, we should be able to predict the second word (Marcheggiani and Titov, 2016; Simon et al., 2019).

2.3 Language Model

Word embeddings are independent from the context, that means the word embedding on words stay same regardless of the sentence they are appeared, and are

hence referred as static embeddings. Due to this limitation, the static embeddings are not capable of recognizing the shift of semantic meaning in different context, such as ambiguity as polysemous words, syntactic structures and semantic roles. In contrary, LMs compute the word embeddings by aggregating the feature from the context, which are hence referred as contextualised embeddings as contrast to the static embeddings. The contextualised embeddings are sensitive to the contexts where a word occurs, and it allows the same word to have different representations depending on the context where the word occurs. Same as word embedding, LMs have the problem of out-of-vocabulary, and it is resolved by using Convolutional Neural Network (CNN) (Jozefowicz et al., 2016; McCann et al., 2017) in early works, and recent LMs rely on Byte Pair Encoding (BPE) (Sennrich et al., 2016) or sentence-piece (Kudo and Richardson, 2018) that are learnt module to tokenize the sentence into sub-words.

2.3.1 Causal Language Model

In the early work, LMs had been studied as an extension of n-gram LMss, that predict the next word in a sentence given a part of the preceding words in the sentence (Jozefowicz et al., 2016), and combined with the pre-trained static embeddings to create the output distribution for words. The models relied on a sequence encoding modules such as Recurrent Neural Network (RNN) or LSTM, followed by softmax activation to represent temporal sequence with a hidden vector, as described in Figure 2.2, that were then used as a feature to solve language understanding tasks (McCann et al., 2017; Peters et al., 2018a; Akbik et al., 2018). More recent works of LMs train LM to learn the contextualised representation from scratch without word embeddings, such as GPT (Generative Pre-trained Transformer) (Radford

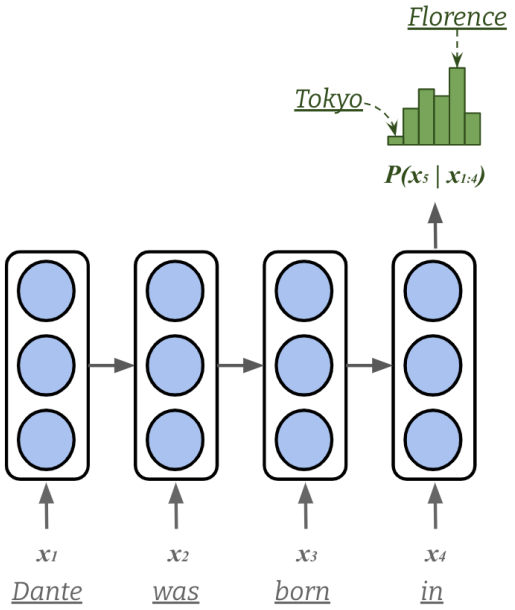

Figure 2.2: An illustration of a CLM performing next token prediction.

et al., 2019; Brown et al., 2020), that employs transformer decoder (Vaswani et al., 2017) rather than LSTM or RNN. The traditional sequence encoding modules such as LSTM or RNN have to process a single token in the input sequence once each time, that slows down the inference in a long sequence. Transformer relies on fully connected layer, which can compute the contextualised representation at once efficiently. It has been shown that transformer is more effective, especially when the model gets scaled (Brown et al., 2020). Nowadays, these type of LM is formally called Causal Language Model (CLM) to emphasize the difference from Masked Language Model (MLM).

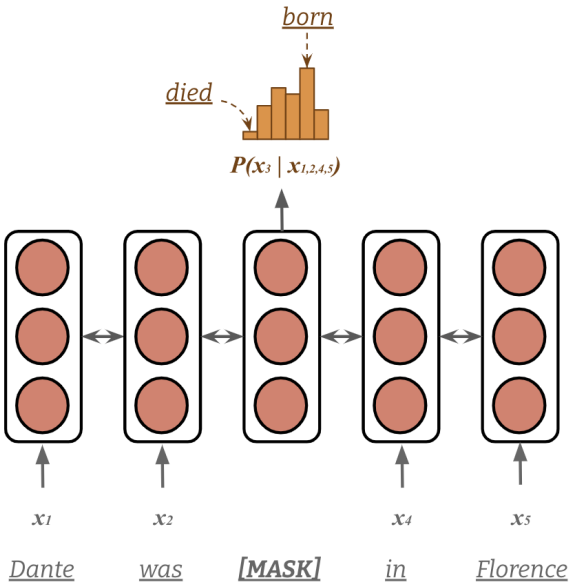

Figure 2.3: An illustration of a MLM performing masked token prediction.

2.3.2 Masked Language Model

BERT (Bidirectional Encoder Representations from Transformers) is another type of LM proposed by Devlin et al. (2019). The key idea of BERT is to use bidirectional transformer encoder unlike unidirectional transformer decoder used in GPT. The bidirectional transformer encoder allows the model to take the left and right contexts of tokens into account at learning the contextualised representation. BERT is trained with MLM pre-training objective, inspired by the Cloze task (Taylor, 1953). Given an input sentence, we choose a word to mask and MLM predicts the masked word from the contextualised representation, as described in Figure 2.3. BERT relies on random masking. In addition to MLM, BERT uses the next sentence prediction task to jointly learn text-pair representations. RoBERTa (Robustly optimized BERT approach) introduced by Liu et al. (2019) is another MLM that follows the same architecture as BERT, but without the next sentence prediction

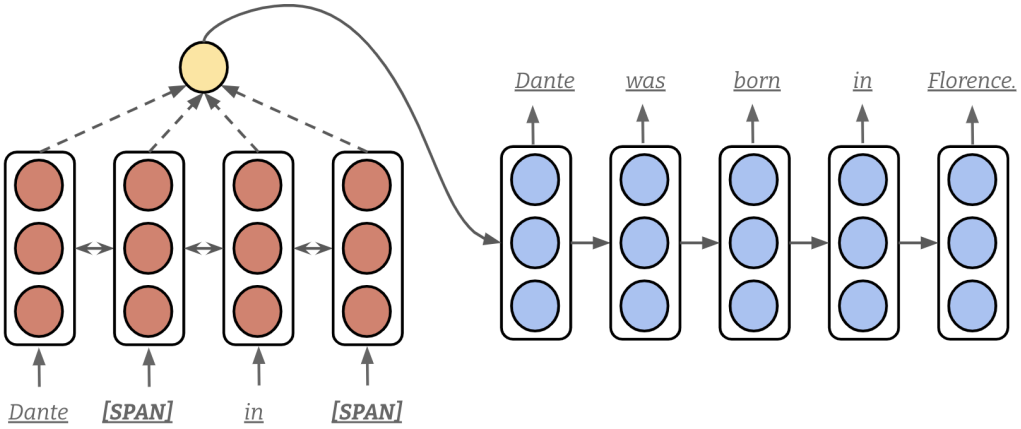

Figure 2.4: An illustration of an EDLM performing span reconstruction.

task and with larger batches over more data. In addition to those improvements, RoBERTa employs a dynamic masking, where the target words to mask are changed over epochs, while BERT relies on static masking.

2.3.3 Encoder-Decoder Language Model

As a comparison of MLM and autoregressive LM, MLMs have a capability to encode information from wider context (i.e. from left and right) into the contextualised representation, because of the transformer encoder, but MLM cannot directly be used for Natural Language Generation (NLG) task such as dialogue, abstractive summarisation, or translation, because MLM is not trained for predicting next words given a context. Meanwhile, although autoregressive LMs limit the information from the previous input at learning contextualised representation, they can be used in any NLP tasks including NLG. To combine the benefit of the capability of MLM to encode the bidirectional information and the ability of strong generative aspect of autoregressive LM, Song et al. (2019) has introduced a unified framework

to pre-train Encoder-Decoder Language Model (EDLM) inspired by the sequence-to-sequence learning (Sutskever et al., 2014). EDLM consists of two components: encoder and decoder, where the encoder relies on bidirectional transformer to extract the semantic from the input sequence in both direction, while the decoder generates tokens one-by-one in an autoregressive manner conditioned on the contextualised feature from the encoder. The encoder and decoder are analogous to MLM and autoregressive LM, where MLM extracts feature from the input, and autoregressive LM outputs text based on the input. MASS (Song et al., 2019) trains the EDLM on the denoising auto-encoder (Lample et al., 2018), which collapses a few spans in the target sentence and predict the complete sentence, as described in Figure 2.4. In addition to the denoising auto-encoder, Lewis et al. (2020a) proposed BART, a EDLM pre-trained on a various set of tasks including sentence permutation, document rotation, and token deletion, which outperformed other EDLM on downstream NLG tasks. Apart from the modifications on the loss objective functions, Raffel et al. (2020) opened a new learning paradigm of text-to-text for EDLM pre-training, and T5, pre-trained EDLM model in text-to-text scheme, that achieves not only competitive results in NLG, but also competitive performance in discriminative task such as document classification as MLM.

2.4 General Knowledge Probing of Language Model

LMs are essentially black box models. While it is clear that they capture an abundance of knowledge, how this knowledge is stored is a matter of ongoing debate. Rather than directly inspecting the parameters of the model, most works there-

fore rely on probing tasks to analyse what the model knows. For instance, a large number of works have focused on analysing the extent to which LMs capture syntax (Goldberg, 2019; Saphra and Lopez, 2019; Hewitt and Manning, 2019; van Schijndel et al., 2019; Jawahar et al., 2019; Tenney et al., 2019), lexical semantics (Ethayarajh, 2019; Bommasani et al., 2020; Vulic et al., 2020), and various forms of factual and commonsense knowledge (Petroni et al., 2019a; Forbes et al., 2019; Davison et al., 2019b; Zhou et al., 2020; Talmor et al., 2020; Roberts et al., 2020). The standard approach of the LM probing relies on a classifier, which is trained to predict some feature of interest from the internal representation of the model. In the case of large LMs with tens of billions of parameters (Wang and Komatsuzaki, 2021; Zhang et al., 2022; Chowdhery et al., 2022; Chung et al., 2022; Tay et al., 2023; Brown et al., 2020; Hoffmann et al., 2022; Rae et al., 2021), probing analyses mainly rely on inspecting the output of the model itself. For instance, there has been considerable interest in the idea that LMs acquire certain abilities when scaled beyond a given point. This is often referred to as *emergent abilities* (Wei et al., 2022). Most relevant to this paper, (Webb et al., 2022) argue that GPT-3 (Brown et al., 2020) obtains the emergent ability to solve a wide range of analogy problems. In the following section, we focus on the relation knowledge specifically, that is one of the understudied aspects for LMs.

2.5 Language Model for Relational Knowledge

The idea of extracting relational knowledge from pre-trained LMs has been extensively studied. For instance, Petroni et al. (2019a) uses BERT for link prediction.

They use a manually defined prompt for each relation type, in which the tail entity is replaced by a `<mask>` token. To complete a knowledge graph triple such as (*Dante*, *born-in*, ?) they create the input "*Dante was born in* `<mask>`" and then look at the predictions of BERT for the masked token to retrieve the correct answer. The results of this analysis suggest that BERT captures a substantial amount of factual knowledge, a finding which has inspired a line of work in which LMs are viewed as knowledge bases. Later, the analysis from Petroni et al. (2019a) has been improved by adding instances with negation in Kassner and Schütze (2020), and extended to non-English languages in Kassner et al. (2021). Some works have also looked at how relational knowledge is stored. In Geva et al. (2020), it is argued that the feed-forward layers of transformer-based LMs act as neural memories, which would suggest that e.g. "the place where Dante is born" is stored as a property of Florence. Some further evidence for this view is presented in Dai et al. (2021). What is less clear is whether relations themselves have an explicit representation, or whether transformer models essentially store a propositionalised knowledge graph. The results we present in this paper suggest that common lexical relations (e.g. hypernymy, meronymy, has-attribute), at least, must have some kind of explicit representation, although it remains unclear how they are encoded. In Bouraoui et al. (2020), they analyse the ability of BERT to identify word pairs that belong to a given relation. In our earlier work Ushio et al. (2021), we have evaluated the ability of LMs to directly solve analogy questions. The main finding was that LMs are poor at solving analogy questions with a vanilla perplexity based approach, although results can be improved with a carefully-tuned scoring function. In Rezaee and Camacho-Collados (2022), they extended this analysis by evaluating the sensitivity of language models to the direction of a word pair (e.g. by checking whether the model can distinguish the word pair *London*:*U.K.* from the word pair *U.K.*:*London*), the ability to recognize which entity type can form a specific

relation type (e.g. the head and tail entity of the *born-in* relation should be person and location) and the robustness to some adversarial examples. Their main findings were that LMs are capable of understanding the direction and the type of a relationship, but can be distracted by simple adversarial examples. For instance, both *Paris*:*France* and *Rome*:*France* were predicted to be instances of the *capital-of* relation.

Given the observation that LMs capture an extensive amount of relational knowledge, LMs have been used for tasks such as KG completion, and even for generating KGs from scratch. For instance, in Davison et al. (2019a), a triple is first converted into a sentence, by choosing a template bases on the log-likelihood estimates of an autoregressive LM. The resulting sentence is then fed into a masked LM to estimate the plausibility of the triple based on the log-likelihood of the masked token prediction on the head and tail words. However, this approach is inefficient to use in practice, since all the candidate triples have to be tested one by one. To avoid such issues, Wang et al. (2020) proposed to directly extract plausible triples using a pre-trained LM. Given a large corpus such as Wikipedia, they parse every sentence in the corpus to find plausible triples with a pre-trained LM. First, a single sentence is fed to a LM to obtain the attention matrix, and then for every combination of two words in the sentence, they find intermediate tokens in between the two words, that contribute to predict the two words by decoding the attention matrix. In the end, the word pairs are simply filtered by the score of the attention based on a threshold, and the resulting word pairs become the triples extracted from the sentence, where the intermediate tokens of each pair is regarded as the relationship between the word pair. Instead of extracting triples from corpus, Alivanistos et al. (2022) proposed to use LMs to complete a triple by generating a tail word given a head word and a relation. They manually create a couple of templates for

each relation type, where a single template contains a placeholder to be filled by a head word. Each template is fed to a pre-trained LM to predict the tail word. As a post-filtering, they use a pre-trained LM to score the factual validity of the generated triples with another prompt to enhance the precision. Unlike the method proposed in Wang et al. (2020) that extracts an entire triple, Alivanistos et al. (2022) assumes the head and the relation are given, so it is more suited to KG completion, while Wang et al. (2020) is rather useful in KG construction on a corpus from scratch. Recently, Huang et al. (2022) proposed a a two-step process for learning a KG in which relations correspond to text descriptions. In the first step, sentences in Wikipedia that explicitly describe relations between entities are identified. To improve the coverage of the resource, in the second step, T5 (Raffel et al., 2020) is used to introduce additional links. Specifically, they use a fusion-in-decoder (Izacard and Grave, 2021) to generate descriptions of the relationship between two entities, essentially by summarising the descriptions of the paths in the original KG that connect the two entities.

Where the aforementioned works extract KGs from LMs, conversely, there has also been a considerable amount of work on infusing the knowledge of existing KGs into LMs. Early approaches introduced auxiliary tasks that were used to train the LM alongside the standard language modelling task, such as entity annotation (Logan et al., 2019) and relation explanation (Hayashi et al., 2020) based on KGs. ERNIE (Sun et al., 2019) is a masked LM similar to BERT, but they employ a masking strategy that focus on entities that are taken from a KG, unlike BERT that randomly mask tokens at pre-training. In addition to the entity-aware masking scheme, LUKE (Yamada et al., 2020) condition internal self-attention by entity-types. They achieved better results than vanilla LMs in many downstream tasks. Since it is computationally demanding to train LMs from scratch, there is another

line of works that rely on fine-tuning existing LMs. For instance, Peters et al. (2019) fine-tuned BERT based on the cross-attention between the embeddings from BERT and an entity linking model. Their model learned a new projection layer to generate entity-aware contextualized embeddings.

2.6 Modelling Analogy

Modelling analogies has a long tradition in the NLP community. The aforementioned LRA model (Turney, 2005), for instance, was motivated by the idea of solving multiple-choice analogy questions. Despite its simplicity, the LRA achieved a strong performance on the SAT benchmark, which even GPT-3 is a not able to beat in the zero-shot setting (Brown et al., 2020). The idea of using word vector differences for identifying analogies was popularised by Mikolov et al. (2013a). The core motivation of using word embeddings for modelling analogies dates back to connectionism theory (Feldman and Ballard, 1982), where neural networks were thought to be capable of learning emergent concepts (Hopfield, 1982; Hinton, 1986) with distributed representations across a semantic embedding space (Hinton et al., 1986). More recent works have proposed mathematical justifications and experiments to understand the analogical capabilities of word embeddings, by attempting to understand their linear algebraic structure (Arora et al., 2016; Gittens et al., 2017; Allen and Hospedales, 2019) and by explicitly studying their compositional nature (Levy and Goldberg, 2014a; Paperno and Baroni, 2016; Ethayarajh et al., 2019; Chiang et al., 2020).

On the other hand, there are criticisms toward the use of word analogy as an evaluation measurement of computational model. In Levy and Goldberg (2014b), it has

been shown that traditional sparse representations over large corpus could obtain the relational similarity by comparing the vector offsets, so the word analogy could not capture the rich semantics of distributed dense representation such as word embedding models. As another specific example, word embedding models are known to be equipped with the linguistic regularities (Mikolov et al., 2013a) but a comprehensive set of experiments have revealed weak correlation of such linguistic regularities and the accuracy on the word analogy (Schluter, 2018).

As a remedy for such limitation in word analogy, Gladkova et al. (2016) pointed out that the main bottleneck in traditional word analogy was raised by the use of a single benchmark, Google Analogy Dataset (Mikolov et al., 2013a), and proposed a new suit of analogy benchmark that contains more diverse linguistic characteristics. In Linzen (2016), they have studied the distance function used to compute relational similarity, which has shown that the commonly used cosine similarity could lead irrelevant neighbourhood in the embedding space, and proposed a new distance function to fix the evaluation protocol of word analogy.

Recently, the focus has shifted to modelling analogies using LMs. For instance, Chen et al. (2022a) proposed E-KAR, a benchmark for analogy modelling which essentially follows the same multiple-choice format as SAT, except that an explanation is provided for why the analogy holds and that some instances involve word triples rather than word pairs. In addition to the task of solving analogy questions, they also consider the task of generating explanations for analogies. Both tasks were found to be challenging for LMs. In Bhavya et al. (2022), they used prompt engineering to generate analogies with GPT-3. They consider two analogy generation tasks: (i) generating an explanation with analogies on a target concept such as "Explain Bohr's atomic model using an analogy", and (ii) generating an explana-

tion of how two given concepts are analoguous to each other such as "Explain how Bohr's atomic model is analogous to the solar system". They argue that GPT-3 is capable of both generating and explaining analogies, but only if an optimal prompt is chosen, where they found the performance to be highly sensitive to the choice of prompt. In Sultan and Shahaf (2022), they used LMs to find analogies between the concepts mentioned in two documents describing situations or processes from different domains. To improve the quality of analogies generated by LMs, Bhavya et al. (2023) proposed an LM based scoring function to detect low-quality analogies. They start from manually-crafted templates that contain the information of the domain (eg. "Machine Learning") and the target concept (eg. "Language Model"). The templates are designed so that LMs can generate explanations to the target concept in the domain with analogies. Once they generate analogies from LMs with the templates, they evaluate the generated analogies from the perspective of analogical style, meaningfulness, and novelty, to identify the analogies with high quality to keep. The evaluation on the analogies are then used to improve the templates, and those low-quality analogies are re-generated with the improved templates. The evaluation relies on automatic metrics, and the template re-writing is done via prompting to edit the current template with the feedback, so the process can be iterate until the low-quality analogies get improved. In Li et al. (2023), they fine-tuned masked LMs to solve analogy question. Given a word pair, an embedding for the word pair is obtained as the embedding of the `<mask>` token with the prompt "word 1 `<mask>` word 1". To fine-tune LMs on analogy question, they convert the task into a binary classification of $A{:}B{::}C{:}D$ is an analogy or not, where A and B is the query word pair and C and D is a candidate word pair. With the binary analogy classification formulation, they fine-tune LMs with a liner layer on top of the word pair embeddings of query and candidate word pairs. After the fine-tuning, they use the fine-tuned model to annotate more instances as data augmen-

tation and continue to fine-tune the model on the generated pseudo dataset. In this thesis, we use analogy as the one of the main suits of evaluating relational knowledge for LMs for the sake of its simplicity yet difficulty, which are further explained in the next chapter Chapter 3.

3 Word Analogies

3.1 Introduction

Analogies play a central role in human commonsense reasoning. The ability to recognize analogies such as "eye is to seeing what ear is to hearing", sometimes referred to as analogical proportions, shape how we structure knowledge and understand language. Surprisingly, however, the task of identifying such analogies has not yet received much attention in the LM era. Given the central role of analogy in human cognition, it is nonetheless important to understand the extent to which NLP models are able to solve these more abstract analogy problems. Besides its value as an intrinsic benchmark for lexical semantics, the ability to recognize analogies is indeed important in the contexts of human creativity (Holyoak et al., 1996), innovation (Hope et al., 2017), computational creativity (Goel, 2019) and education (Pardos and Nam, 2020). Analogies are also a prerequisite to build Artificial Intelligence (AI) systems for the legal domain (Ashley, 1988; Walton, 2010) and are used in machine learning (Miclet et al., 2008; Hug et al., 2016; Hüllermeier, 2020) and

for ontology alignment (Raad and Evermann, 2015), among others.

In this chapter, we introduce the analogy question, a common benchmark of modelling analogy in § 3.2. Analogy question is a multiple-choice Question Answering (QA) task, where the query and the choices are all word pairs. We compile five diverse analogy question datasets in a unified format. We then explain commonly used baselines to solve the analogy questions with a statistical approach, static word embeddings, and LM in § 3.3. We then introduce scoring functions tailored for analogy questions in § 3.4. The aim in this chapter is to analyze the ability of pre-trained LMs to recognize analogies with simple baselines and scoring functions tailored for analogy questions. We extensively analyze the impact of both of these choices, as well as the differences between different LMs. We show that the analogy question is a highly demanding task for LMs and most of the large LM cannot even outperform a static word embedding, or statistical baseline in the zero-shot setting. However, when the prompt and scoring function are carefully calibrated, we find that GPT-2 can outperform the SoTA statistical baseline, standard word embeddings as well as the published results for GPT-3 in the zero-shot setting. However, we also find that these results are highly sensitive to the choice of the prompt, as well as two hyperparameters in our scoring function, with the optimal choices not being consistent across different datasets. Moreover, using BERT leads to considerably weaker results, underperforming even standard word embeddings in all of the considered configurations.

Query:		wing:air
Candidates:	(1)	arm:hand
	(2)	lung:breath
	(3)	**flipper:water**
	(4)	cloud:sky
	(5)	engine:jet

Table 3.1: An example of analogy task from the SAT dataset, where the candidate in bold characters is the answer in each case.

Dataset	Avg. #Answer Candidates	#Questions
SAT	5	- / 374
U2	4	24 / 228
U4	4	48 / 432
Google	4	50 / 500
BATS	4	199 / 1,799

Table 3.2: Main statistics of the analogy question datasets, showing the average number of answer candidates, and the total number of questions (validation / test).

3.2 Analogy Question Datasets

Analogy questions are multiple-choice questions, where a given word pair is provided, called the *query pair*, and the task is to predict which among a number of candidate answers is the word pair that is most analogous to the query pair (see an example at Table 3.1). In other words, the task is to find the word pair whose relationship best resembles the relationship between the words in the query. We use the following five datasets, where Table 3.2 summarises the main features of the analogy question datasets, and Table 3.3 shows an example from each of the analogy questions[1]. Importantly, none of the datasets contain training set, so the task is an

[1]Preprocessed versions of the datasets are available at https://huggingface.co/datasets/relbert/analogy_questions except for SAT, which is not publicly released yet.

Dataset	Domain	Example
SAT	College Admission Test	[beauty, aesthete, pleasure, hedonist]
U2	Grade4	[rock, hard, water, wet]
	Grade5	[hurricane, storm, table, furniture]
	Grade6	[microwave, heat, refrigerator, cool]
	Grade7	[clumsy, grace, doubtful, faith]
	Grade8	[hidden, visible, flimsy, sturdy]
	Grade9	[panacea, cure, contagion, infect]
	Grade10	[grain, silo, water, reservoir]
	Grade11	[thwart, frustrate, laud, praise]
	Grade12	[lie, prevaricate, waver, falter]
U4	Low Intermediate	[accident, unintended, villain, evil]
	Low Advanced	[galleon, sail, quarantine, isolate]
	High Beginning	[salesman, sell, mechanic, repair]
	High Intermediate	[classroom, desk, church, pew]
	High Advanced	[erudite, uneducated, fervid, dispassionate]
BATS	Inflectional Morphology	[neat, neater, tasty, tastier]
	Derivational Morphology	[available, unavailable, interrupted, uninterrupted]
	Encyclopedic Semantics	[stockholm, sweden, belgrade, serbia]
	Lexicographic Semantics	[elephant, herd, flower, bouquet]
Google	Encyclopedic Semantics	[Canada, dollar, Croatia, kuna]
	Morphological	[happy, happily, immediate, immediately]

Table 3.3: An example from each domain of the analogy question benchmarks.

unsupervised task. Please ask to the author of Turney et al. (2003) to obtain SAT dataset.

3.2.1 SAT

The SAT exam is a US college admission test. Turney et al. (2003) collected a benchmark of 374 word analogy problems, consisting primarily of problems from these SAT tests. Each instance has five candidates. The instances are aimed at college applicants, and are thus designed to be challenging for humans. Aimed at college

applicants, these problems are designed to be challenging for humans.

3.2.2 U2&U4

Following Boteanu and Chernova (2015), who used word analogy problems from an educational website[2], we compiled analogy questions from the same resource[3]. They used in particular UNIT 2 of the analogy problems from the website, which have the same form as those from the SAT benchmark, but rather than college applicants, they are aimed at children in grades 4 to 12 from the US school system (i.e. from age 9 onwards). We split the dataset into 24 questions for validation and 228 questions for testing. Each question has 4 answer candidates. We have collected another benchmark from the UNIT 4 problems on the same website that was used for the U2 dataset. These UNIT 4 problems are organised in 5 difficulty levels: high-beginning, low-intermediate, high-intermediate, low-advanced and high-advanced. The low-advanced level is stated to be at the level of the SAT tests, whereas the high-advanced level is stated to be at the level of the GRE test (which is used for admission into graduate schools). The resulting U4 dataset has 48 questions for validation and 432 questions for testing. Each question has 4 answer candidates.

3.2.3 Google

The Google analogy dataset (Mikolov et al., 2013b) has been one of the most commonly used benchmarks for evaluating word embeddings[4]. This dataset contains a

[2] https://www.englishforeveryone.org/Topics/Analogies.html

[3] We use the dataset from the website with permission limited to research purposes.

[4] The original data is available at https://aclweb.org/aclwiki/Google_analogy_test_set_(State_of_the_art)#cite_note-1.

mix of semantic and morphological relations such as *capital-of* and *singular-plural*, respectively. The dataset was tailored to the evaluation of word embeddings in a predictive setting. We constructed word analogy problems from the Google dataset by choosing for each correct analogy pair a number of negative examples. To obtain sufficiently challenging negative examples, for each query pair (e.g. *Paris-France*) we extracted three negative instances:

1. two random words from the head of the input relation type (e.g. *Rome-Oslo*);
2. two random words from the tail of the input relation type (e.g. *Germany-Canada*);
3. a random word pair from a relation type of the same high-level category (i.e. semantic or morphological) as the input relation type (e.g. *Argentina-peso*).

The resulting dataset contains 50 validation and 500 test questions, each with 4 answer candidates.

3.2.4 BATS

The coverage of Google dataset is known to be limiting, and BATS (Gladkova et al., 2016) was developed in an attempt to address its main shortcomings. BATS[5] includes a larger number of concepts and relations, which are split into four categories: lexicographic, encyclopedic, and derivational and inflectional morphology. We follow the same procedure as for the Google dataset to convert BATS into the

[5]The original data is available at https://vecto.space/projects/BATS/.

analogy question format. BATS contains 199 validation and 1,799 test questions, each with 4 answer candidates.

3.3 Baselines

We now introduce the baselines we considered for solving analogy questions. As the datasets do not contain any task specific training dataset, all the baselines should be unsupervised method, and for the simplicity, we focus on baselines in the zero-shot setting, which do not involve any tuning of the model on the validation set.

3.3.1 Random Baseline

First, we consider the expected accuracy over a random prediction, which we refer as the *random baseline*. The random baseline on a analogy question is simply computed by averaging the inverse of the number of the candidate.

3.3.2 Latent Relation Analysis

LRA takes inspiration from the Latent Semantic Analysis (LSA) model for learning document embeddings (Deerwester et al., 1990). The key idea behind LSA was to apply Singular Value Decomposition (SVD) on a document-term co-occurrence matrix to obtain low-dimensional vector representations of documents. LRA similarly uses SVD to learn relation vectors. In particular, the method also constructs a co-occurrence matrix, where the rows now correspond to word pairs and the columns

correspond to lexical patterns. Each matrix entry captures how often the corresponding word pair appears together with the corresponding lexical pattern in a given corpus. To improve the quality of the representations, a Point-wise Mutual Information (PMI)-based weighting scheme is used, and the method also counts occurrences of synonyms of the words in a given pair. To solve an analogy question, we can compute the LRA embeddings of the query pair and the candidate pairs, and then select the answer whose embedding is closest to the embedding of the query pair, in terms of cosine similarity. Since the official implementation of LRA as well as the corpus used to compute the low-dimensional vector representations are not publicly released, we report the published results from Turney (2005) for the SAT dataset only.

3.3.3 Word Embedding

Since the introduction of the Word2Vec models (Mikolov et al., 2013a), word analogies have been a popular benchmark for evaluating different word embedding models. This stems from the observation that in many word embedding models, the relation between two words A and B is to some extent captured by the vector difference of their embeddings. Writing $\mathsf{wv}(A)$ for the word embedding of a word A, we can thus learn relation embeddings of the form $\mathbf{x}_{(A,B)} = \mathsf{wv}(B) - \mathsf{wv}(A)$. Using these embeddings, we can solve word analogy questions by again selecting the answer candidate whose embedding is most similar to that of the query pair, in terms of cosine similarity. Since some of the analogy questions include rare words, common word embedding models such as word2vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) suffer from the out-of-vocabulary issue. We therefore use fastText (Bojanowski et al., 2016) trained on Common Crawl with subword infor-

mation, which can handle out-of-vocabulary words by splitting them into smaller chunks of characters[6].

3.3.4 Language Models

To solve analogy questions using a pre-trained LM, we proceed as follows. Let (A, B) be the query pair. For each answer candidate (C, D) we construct a sentence with a manual template. Following Brown et al. (2020), we use

> A is to B what C is to D

as a default. We then compute the perplexity of each of these sentences, and predict the candidate that gives rise to the lower perplexity. The exact computation depends on the type of language model that is considered. For CLMs, the perplexity of a sentence $\boldsymbol{s}$ can be computed as follows:

$$f(\boldsymbol{x}) = \exp\left(-\frac{1}{m}\sum_{j=1}^{m}\log P_{\text{clm}}(x_j|\boldsymbol{x}_{j-1})\right) \tag{3.1}$$

where $\boldsymbol{s}$ is tokenized as $[x_1...x_m]$ and $P_{\text{clm}}(x|\boldsymbol{x})$ is the likelihood from an CLM's next token prediction.

For MLM, we instead use pseudo-perplexity (Salazar et al., 2020), which is defined as

$$f(\boldsymbol{x}) = \exp\left(-\frac{1}{m}\sum_{j=1}^{m}\log P_{\text{mask}}(x_j|\boldsymbol{x}_{\setminus j})\right) \tag{3.2}$$

[6]The embedding model is available at `https://fasttext.cc/`.

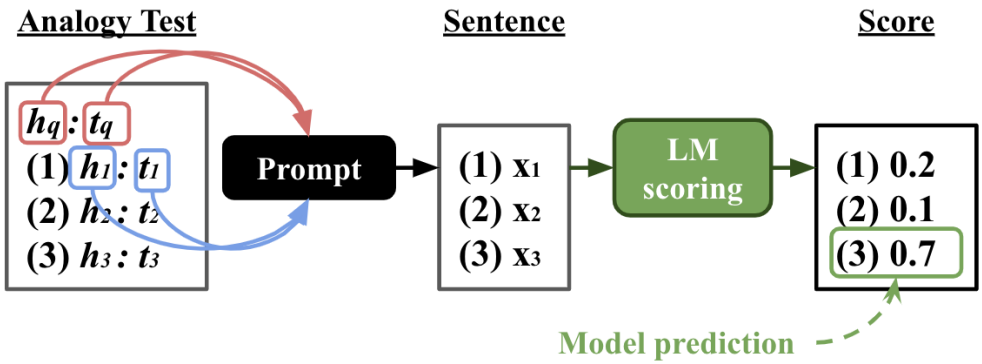

Figure 3.1: Solving a word analogy problem by selecting one with the highest LM score among the candidates.

where

$$\boldsymbol{x}_{\setminus j} = [x_1 \dots x_{j_1} \langle \text{mask} \rangle x_{j+1} \dots x_m] \tag{3.3}$$

and $P_{\text{mask}}(x_j|\boldsymbol{x}_{\setminus j})$ is the pseudo-likelihood (Wang and Cho, 2019) that the masked token is x_j.

Finally, for EDLMs, we split the template in two parts. For example, with the default template, the phrase "A is to B" is fed into the encoder, and then we use the decoder to compute the perplexity of the phrase "C is to D", using the probability P_{clm} of the decoder, conditioned by the encoder.

3.4 Analogical Proportion Score

In this section, we explain our strategy for using pre-trained LMs to solve analogy questions without fine-tuning to go beyond the simple zero-shot perplexity based method we described in § 3.3.4. To be precise, we generalize perplexity to be a scoring function that produces a validity score to each candidate given the query word

pair based on Analogical Proportion (AP), and hence we refer the scores as AP score.

First, in § 3.4.1 we explain how each relation pair is converted into a natural sentence to be fed into the LM. In § 3.4.2, we then discuss a number of scoring functions that can be used to select the most plausible answer candidate. Finally, we take advantage of the fact that AP is invariant to particular permutations, which allows for a natural extension of the proposed scoring functions (§ 3.4.3). Figure 3.1 shows a high-level overview of our methodology. Importantly, the perplexity baseline described in § 3.3.4 is a special case of the AP score, where the scoring function is perplexity.

Note that the templates are different from the schema of KG, and they do not represent any specific relation type. The prompt is to represent any type of relationship among the four words.

3.4.1 Relation Pair Prompting

In § 3.3.4, we use the analogical statement as the template to transform the query pair (A, B) and an answer candidate (C, D) following Brown et al. (2020). For the flexibility to search an optimal template for each LM, we define a prompting function $\mathcal{T}_t(A, B, C, D)$ that takes four placeholders and a template type t, and returns a sentence in which the placeholders were replaced by the words A, B, C, and D. For instance, given a query "word:language" and a candidate "note:music", the prompting function produces

$\mathcal{T}_{to\text{-}as}($"word", "language", "note", "music"$) =$

Type	Template
to-as	[A] is to [B] as [C] is to [D]
to-what	[A] is to [B] What [C] is to [D]
rel-same	The relation between [A] and [B] is the same as the relation between [C] and [D].
what-to	what [A] is to [B], [C] is to [D]
she-as	She explained to him that [A] is to [B] as [C] is to [D]
as-what	As I explained earlier, what [A] is to [B] is essentially the same as what [C] is to [D].

Table 3.4: Custom templates used in our experiments. Each has four placeholders $[A, B, C, D]$ and they are fulfilled by words from a relation pair.

"word is to language as note is to music" (3.4)

where we use the template type *to-as* here. Table 3.4 shows the set of custom templates we considered in our experiment. The template *to-what* is the default template for the zero-shot LM baselines in § 3.3.4. Using manually specified template types can result in a sub-optimal textual representation. For this reason, recent studies have proposed auto-prompting strategies, which optimize the template type on a training set (Shin et al., 2020), paraphrasing (Jiang et al., 2020), additional prompt generation model (Gao et al., 2020), and corpus-driven template mining (Bouraoui et al., 2020). However, none of these approaches can be applied to unsupervised settings. Thus, we do not explore auto-prompting methods in this work. Instead, we will consider a number of different template types in the experiments, and assess the sensitivity of the results to the choice of template type.

3.4.2 Scoring Function

In this section, we explain the new scoring functions, we proposed to solve analogy questions with LMs. The scoring functions are all based on the zero-shot perplexity baseline (§ 3.3.4). As a matter of convenience, we frame the analogy task in terms of analogical proportions (Prade and Richard, 2017). Given a query word pair (h_q, t_q) and a list of candidate answer pairs $\{(h_i, t_i)\}_{i=1}^{n}$, the goal is to find the candidate answer pair that has the most similar relation to the query pair.

3.4.2.1 PMI

Perplexity is well-suited to capture the fluency of a sentence, but it may not be the best choice to test the plausibility of a given AP candidate. As an alternative, we propose a scoring function that focuses specifically on words from the two given pairs. To this end, we propose to use an approximation of PMI, based on perplexity. PMI is defined as the difference between a conditional and marginal log-likelihood. In our case, we consider the conditional likelihood of t_i given h_i and the query pair, i.e. $P(t_i|h_q, t_q, h_i)$, and the marginal likelihood over h_i, i.e. $P(t_i|h_q, t_q)$. Subsequently, the PMI-inspired scoring function is defined as

$$r(t_i|h_i, h_q, t_q) = \log P(t_i|h_q, t_q, h_i) - \alpha \cdot \log P(t_i|h_q, t_q) \tag{3.5}$$

where α is a hyperparameter to control the effect of the marginal likelihood. The PMI score corresponds to the specific case where $\alpha = 1$. However, Davison et al. (2019b) found that using a hyperparameter to balance the impact of the conditional and marginal probabilities can significantly improve the results. The prob-

abilities in Equation 3.5 are estimated by assuming that the answer candidates are the only possible word pairs that need to be considered. By relying on this closed-world assumption, we can estimate marginal probabilities based on perplexity, which we found to give better results than the masking based strategy from Davison et al. (2019b). In particular, we estimate these probabilities as

$$P(t_i|h_q, t_q, h_i) = -\frac{f\left(\mathcal{T}_t(h_q, t_q, h_i, t_i)\right)}{\sum_{k=1}^{n} f\left(\mathcal{T}_t(h_q, t_q, h_i, t_k)\right)} \tag{3.6}$$

$$P(t_i|h_q, t_q) = -\frac{\sum_{k=1}^{n} f\left(\mathcal{T}_t(h_q, t_q, h_k, t_i)\right)}{\sum_{k=1}^{n}\sum_{l=1}^{n} f\left(\mathcal{T}_t(h_q, t_q, h_k, t_l)\right)} \tag{3.7}$$

where n is the number of answer candidates for the given query and f is perplexity defined in Equation 3.1 for CLMs or pseudo-perplexity defined in Equation 3.2 for MLMs. Equivalently, since PMI is symmetric, we can consider the difference between the logs of $P(h_i|h_q, t_q, t_i)$ and $P(h_i|h_q, t_q)$. While this leads to the same PMI value in theory, due to the way in which we approximate the probabilities, this symmetric approach will lead to a different score. We thus combine both scores with an aggregation function $\mathcal{A}_g$. This aggregation function takes a list of scores and outputs an aggregated value. As an example, given a list $[1, 2, 3, 4]$, we write $\mathcal{A}_{\text{mean}}([1, 2, 3, 4]) = 2.5$ for the mean and $\mathcal{A}_{\text{val}_1}([1, 2, 3, 4]) = 1$ for the first element. Given such an aggregation function, we define the following PMI-based score

$$s_{PMI}(t_i, h_i|h_q, t_q) = \mathcal{A}_g\left(\boldsymbol{r}\right) \tag{3.8}$$

where we consider basic aggregation operations over the list of

$$\boldsymbol{r} = [r(t_i|h_i, h_q, t_q), r(h_i|t_i, h_q, t_q)] \tag{3.9}$$

such as the mean, max, and min value. The choice of using only one of the scores $r(t_i|h_i, h_q, t_q), r(h_i|t_i, h_q, t_q)$ is viewed as a special case, in which the aggregation function g simply returns the first or the second item.

3.4.2.2 mPPL

We also experiment with a third scoring function, which borrows ideas from both perplexity and PMI. In particular, we propose the Marginal Likelihood biased Perplexity (mPPL) defined as

$$s_{mPPL}(t_i, h_i|h_q, t_q) = \log s_{PPL}(t_i, h_i|h_q, t_q) \tag{3.10}$$

$$- \alpha_t \cdot \log P(t_i|h_q, t_q) \tag{3.11}$$

$$- \alpha_h \cdot \log P(h_i|h_q, t_q) \tag{3.12}$$

where α_t and α_h are hyperparameters, and s_{PPL} is a normalized perplexity defined as

$$s_{PPL}(t_i, h_i|h_q, t_q) = -\frac{f\left(\mathcal{T}_t(h_q, t_q, h_i, t_i)\right)}{\sum_{k=1}^{n} f\left(\mathcal{T}_t(h_q, t_q, h_k, t_k)\right)}. \tag{3.13}$$

The mPPL score extends perplexity with two bias terms. It is motivated from the insight that treating α as a hyperparameter in Equation 3.5 can lead to better results than fixing $\alpha = 1$. By tuning α_t and α_h, we can essentially influence to what extent answer candidates involving semantically similar words to the query pair should be favored.

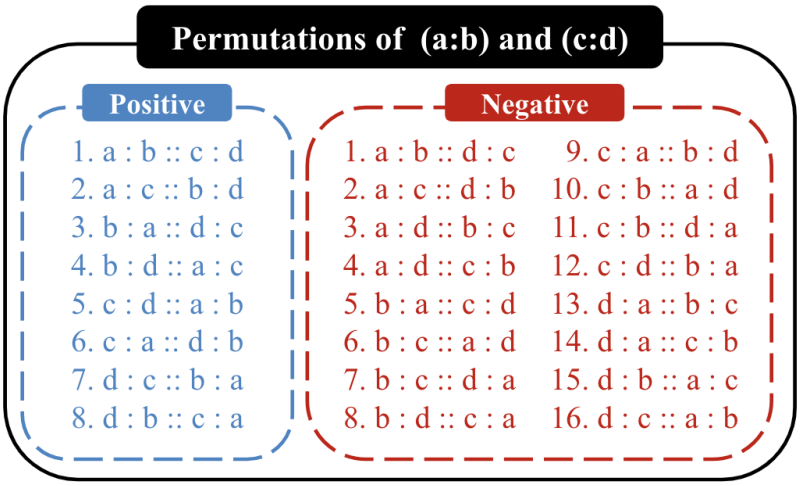

Figure 3.2: Positive and negative permutations for a relation pair *(a:b)-(c:d)*.

3.4.3 Permutation Invariance

The formalization of APs dates back to Aristotle (Barbot et al., 2019). According to the standard axiomatic characterization, whenever we have an AP $a : b :: c : d$ (meaning "a is to b what c is to d"), it also holds that $c : d :: a : b$ and $a : c :: b : d$ are APs. It follows from this that for any given AP $a : b :: c : d$ there are eight permutations of the four elements a, b, c, d that form APs. These eight permutations, along with the 16 "negative permutations", are shown in Figure 3.2. To take advantage of the different permutations of APs, we propose the following AP score:

$$AP(h_q, t_q, h_i, t_i) = \mathcal{A}_{g_{\text{pos}}}(\boldsymbol{p}) - \beta \cdot \mathcal{A}_{g_{\text{neg}}}(\boldsymbol{n}) \tag{3.14}$$

$$\boldsymbol{p} = [s(a, b|c, d)]_{(a:b,c:d)\in \boldsymbol{P}}$$

$$\boldsymbol{n} = [s(a, b|c, d)]_{(a:b,c:d)\in \boldsymbol{N}}$$

Parameter	Value
α	-0.4, -0.2, 0, 0.2, 0.4
α_h	-0.4, -0.2, 0, 0.2, 0.4
α_t	-0.4, -0.2, 0, 0.2, 0.4
β	0, 0.2, 0.4, 0.6, 0.8, 1.0
g	max, mean, min, val_1, val_2
g_{pos}	max, mean, min, val_1, ..., val_8
g_{neg}	max, mean, min, val_1, ..., val_{16}

Table 3.5: The search space of each hyperparameter.

where $\boldsymbol{P}$ and $\boldsymbol{N}$ correspond to the list of positive and negative permutations of the candidate AP $h_q : t_q :: h_i : t_i$ in the order shown in Figure 3.2, β is a hyperparameter to control the impact of the negative permutations, and $s(a, b|c, d)$ is a scoring function as described in § 3.4.2. Here $\mathcal{A}_{g_{\text{pos}}}$ and $\mathcal{A}_{g_{\text{neg}}}$ refer to the aggregation functions that are used to combine the scores for the positive and negative permutations respectively, where these aggregation functions are defined as in § 3.4.2. To solve an analogy problem, we simply choose the answer candidate that results in the highest value of $AP(t_i, h_i, h_q, t_q)$.

3.5 Experimental Setting

We consider three transformer-based LMs of a different nature: two MLM, namely BERT and RoBERTa, and GPT-2, as a prominent example of a CLM. Each pretrained model was fetched from the Huggingface model hub (Wolf et al., 2020), from which we use `bert-large-cased`[7], `roberta-large`[8], and `gpt2-xl`[9] respectively. For parameter selection, we run grid search on β, α, α_h, α_t, t, g, g_{pos}, and

[7] https://huggingface.co/bert-large-cased
[8] https://huggingface.co/roberta-large
[9] https://huggingface.co/gpt2-xl

g_{neg} for each model and select the configuration which achieves the best accuracy on each validation set. We experiment with the three scoring functions presented in § 3.4.2, which we s_{PPL} (perplexity), s_{PMI} and s_{mPPL}. As our scoring functions require to compute the perplexity of all the permutations in every analogy question instance, we focus on the five analogy questions, SAT, U2, U4, BATS, and Google, which have a few candidates. Since the original SAT dataset does not contain a validation set, we randomly sample 10% of the original SAT datasets as a validation set and consider the rest of the dataset as the test set, which we refer as $\text{SAT}_{\text{SPLIT}}$. Please check § 3.2 for more detail about the analogy question datasets. We ran grid search to find the optimal parameters for all the scoring function in each combination of the dataset and the LM, where each parameter was selected within the values shown in Table 3.5. As the coefficient of marginal likelihood $\alpha, \alpha_h, \alpha_t$, we considered negative values as well as we hypothesized that the marginal likelihood could be beneficial for LMs as a way to leverage lexical knowledge of the head and tail words.

3.6 Experimental Results

Table 3.6 shows our main results, where Table 3.8 and Table 3.9 present the best parameters found with the gird search in each scoring function and the LM. As far as the comparison among LMs is concerned, RoBERTa and GPT-2 consistently outperform BERT. Among the AP variants, s_{mPPL} achieves substantially better results than s_{PMI} or s_{PPL} in most cases. We also observe that word embeddings perform surprisingly well, with fastText and GloVe outperforming BERT on most datasets, as well as GPT-2 and RoBERTa with default hyperparameters. Fast-

Model	Score	Tuned	SAT_{SPLIT}	U2	U4	Google	BATS	Average
BERT	s_{PPL}		32.9	36.0	36.6	79.8	59.4	48.9
		✓	39.8	41.7	41.0	86.8	67.9	55.4
	s_{PMI}		27.0	32.0	31.2	74.0	59.1	44.7
		✓	40.4	42.5	27.8	87.0	68.1	53.2
	s_{mPPL}	✓	41.8	44.7	41.2	88.8	67.9	56.9
RoBERTa	s_{PPL}		42.4	50.4	49.8	88.8	69.9	60.2
		✓	53.7	57.0	55.8	93.6	80.5	68.1
	s_{PMI}		35.9	42.5	44.0	60.8	60.8	48.8
		✓	51.3	49.1	38.7	92.4	77.2	61.7
	s_{mPPL}	✓	53.4	**58.3**	**57.4**	93.6	78.4	**68.2**
GPT-2	s_{PPL}		35.9	41.2	44.9	80.4	63.5	53.2
		✓	50.4	48.7	51.2	93.2	75.9	63.9
	s_{PMI}		34.4	43.0	44.0	80.4	62.0	52.7
		✓	51.0	37.7	50.5	91.0	79.8	62.0
	s_{mPPL}	✓	**56.7**	50.9	49.5	95.2	**81.2**	66.7
fastText	-		47.8	38.2	38.4	**94.6**	70.7	57.9
Random	-		20.0	23.6	24.2	25.0	25.0	23.6

Table 3.6: The accuracy on each analogy dataset. All LMs use the AP function described in § 3.4.3. The default configuration for AP includes $\alpha = \alpha_h = \alpha_t = \beta = 0$, $g_{\text{pos}} = g = \text{val}_1$, and $t = \textit{to-as}$. Note that $s_{PPL} = s_{mPPL}$ with the default configuration. The average accuracy across datasets is included in the last column.

Text achieves the best overall accuracy on the Google dataset, confirming that this dataset is particularly well-suited to word embeddings.

In order to compare with published results from prior work, we carried out an additional experiment on the full SAT dataset (i.e., without splitting it into validation and test). Table 3.7 shows the results. GPT-3 and LRA are added for comparison. Given the variability of the results depending on the tuning procedure, we have also reported results of configurations that were tuned on the entire set, to provide an upper bound on what is possible within the proposed unsupervised setting. This result shows that even with optimal hyperparameter values, LMs barely outperform the performance of the simpler LRA model. GPT-3 similarly fails to outperform

	Model	Score	Tuned	The accuracy
LM	BERT	s_{PPL}	✓	32.9 40.4*
		s_{PMI}	✓	26.8 41.2*
		s_{mPPL}	✓	42.8*
	RoBERTa	s_{PPL}	✓	40.6 55.8*
		s_{PMI}	✓	42.5 54.0*
		s_{mPPL}	✓	55.8*
	GPT-2	s_{PPL}	✓	36.9 56.2*
		s_{PMI}	✓	34.7 56.8*
		s_{mPPL}	✓	57.8*
	GPT-3	*Zero-shot* *Few-shot*	✓	*53.7* *65.2**
LRA		-		*56.4*
fastText		-		49.7
Random		-		20.0

Table 3.7: The accuracy results for the full SAT dataset. Results marked with * are not directly comparable as they were tuned on full data (for our models) or use training data (for GPT-3 few-shot). These results are included to provide an upper bound only. The results in italics were taken from the original papers.

LRA in the zero-shot setting.

3.6.1 Prediction Breakdown

To increase our understanding of what makes an analogy problem difficult for LMs, we compare the results for each difficulty level. LMs use s_{mPPL} with the best configuration tuned in the corresponding validation sets. We also note that the number of candidates in U2 and U4 vary from three to five, so results per difficulty level are

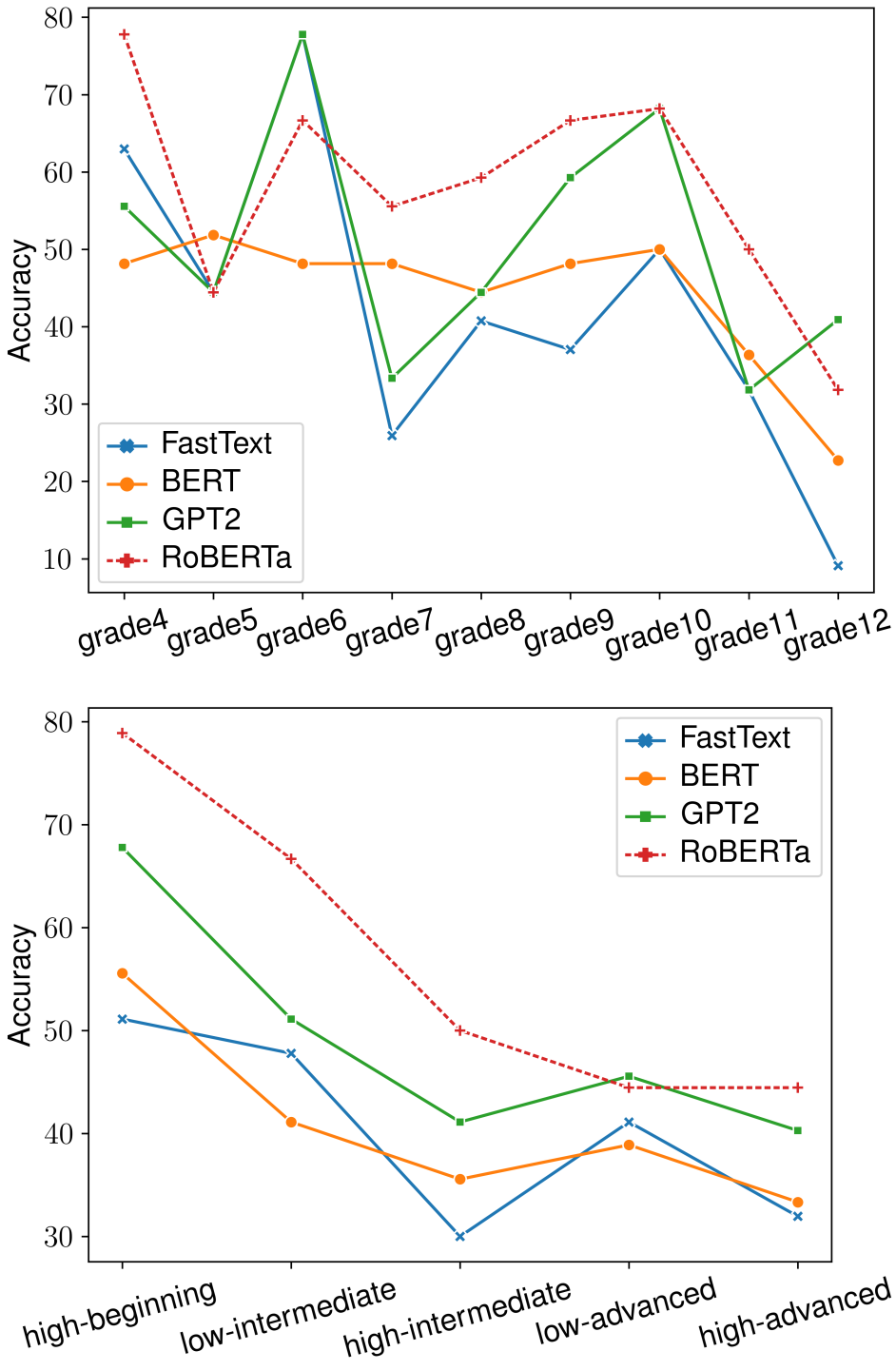

Figure 3.3: The test accuracy in U2 (top) and U4 (bottom) per difficulty level.

	Data	g	α	g_{pos}	g_{neg}	β	t
BERT	$\text{SAT}_{\text{SPLIT}}$	val_2	-0.4	val_5	val_{12}	0.4	*what-to*
	U2	val_2	-0.4	mean	mean	0.6	*what-to*
	U4	val_1	0.4	max	val_7	1.0	*rel-same*
	Google	val_1	-0.4	val_1	val_{11}	0.4	*she-as*
	BATS	val_1	-0.4	val_{11}	val_1	0.4	*she-as*
RoBERTa	$\text{SAT}_{\text{SPLIT}}$	min	-0.4	min	val_7	0.2	*as-what*
	U2	min	0.4	mean	val_4	0.6	*what-to*
	U4	val_2	0.0	mean	val_4	0.8	*to-as*
	Google	val_1	-0.4	val_1	val_6	0.4	*what-to*
	BATS	max	-0.4	mean	val_{11}	0.6	*what-to*
GPT-2	$\text{SAT}_{\text{SPLIT}}$	val_2	-0.4	val_3	val_1	0.6	*rel-same*
	U2	val_2	0.0	val_4	val_4	0.6	*rel-same*
	U4	val_2	-0.4	mean	mean	0.6	*rel-same*
	Google	val_1	0.0	mean	val_{11}	0.4	*as-what*
	BATS	val_1	-0.4	val_1	val_6	0.4	*rel-same*

Table 3.8: The best configuration of s_{PMI} score.

	Data	α_h	α_t	g_{pos}	g_{neg}	β	t
BERT	$\text{SAT}_{\text{SPLIT}}$	-0.2	-0.4	val_5	val_5	0.2	*what-to*
	U2	0.0	-0.2	mean	mean	0.8	*she-as*
	U4	-0.2	0.4	val_7	min	0.4	*to-as*
	Google	0.4	-0.2	val_5	val_{12}	0.6	*she-as*
	BATS	0.0	0.0	val_8	min	0.4	*what-to*
RoBERTa	$\text{SAT}_{\text{SPLIT}}$	0.2	0.2	val_5	val_{11}	0.2	*as-what*
	U2	0.4	0.4	val_1	val_4	0.4	*what-to*
	U4	0.2	0.2	val_1	val_1	0.4	*as-what*
	Google	0.2	0.2	val_1	val_6	0.2	*what-to*
	BATS	0.2	-0.2	val_5	val_{11}	0.4	*what-to*
GPT-2	$\text{SAT}_{\text{SPLIT}}$	-0.4	0.2	val_3	val_1	0.8	*rel-same*
	U2	-0.2	0.2	mean	mean	0.8	*as-what*
	U4	-0.2	0.2	mean	mean	0.8	*rel-same*
	Google	-0.2	-0.4	mean	mean	0.8	*rel-same*
	BATS	0.4	-0.4	val_1	val_5	0.8	*rel-same*

Table 3.9: The best configuration of s_{mPPL} score.

not fully comparable. However, they do reflect the actual difficulty of the educational tests.

Figure 3.3 shows the results of all LMs (tuned setting), fastText and the PMI base-

line according to these difficulty levels. Broadly speaking, we can see that instances that are harder for humans are also harder for the considered models. The analogies in the most difficult levels are generally more abstract (e.g. *witness* : *testimony* :: *generator* : *electricity*), or contain obscure or infrequent words (e.g. *grouch* : *cantakerous* :: *palace* : *ornate*).

Figure 3.4 shows the results of different LMs with the s_{mPPL} scoring function on the different categories of the BATS and Google datasets. We can confirm that LMs are worse than fastText in the semantic relation in Google, where BERT and RoBERTa are poor at morphological types too. Meanwhile, LMs are better than fastText in lexical relation type of BATS dataset. Also, GPT-2 is good at encyclopedic relation type within all baselines.

3.7 Analysis

We now take a closer look into our results to investigate parameter sensitivity, the correlation between model performance and human difficulty levels, and possible dataset artifacts. The following analysis focuses on s_{mPPL} as it achieved the best results among the LM based scoring functions. We first analyze errors made by each LM in § 3.7.1, test the hypothesis only method in § 3.7.2, compare a few more additional scoring functions § 3.7.4, and check the sensitivity to the parameter choice in § 3.7.3.

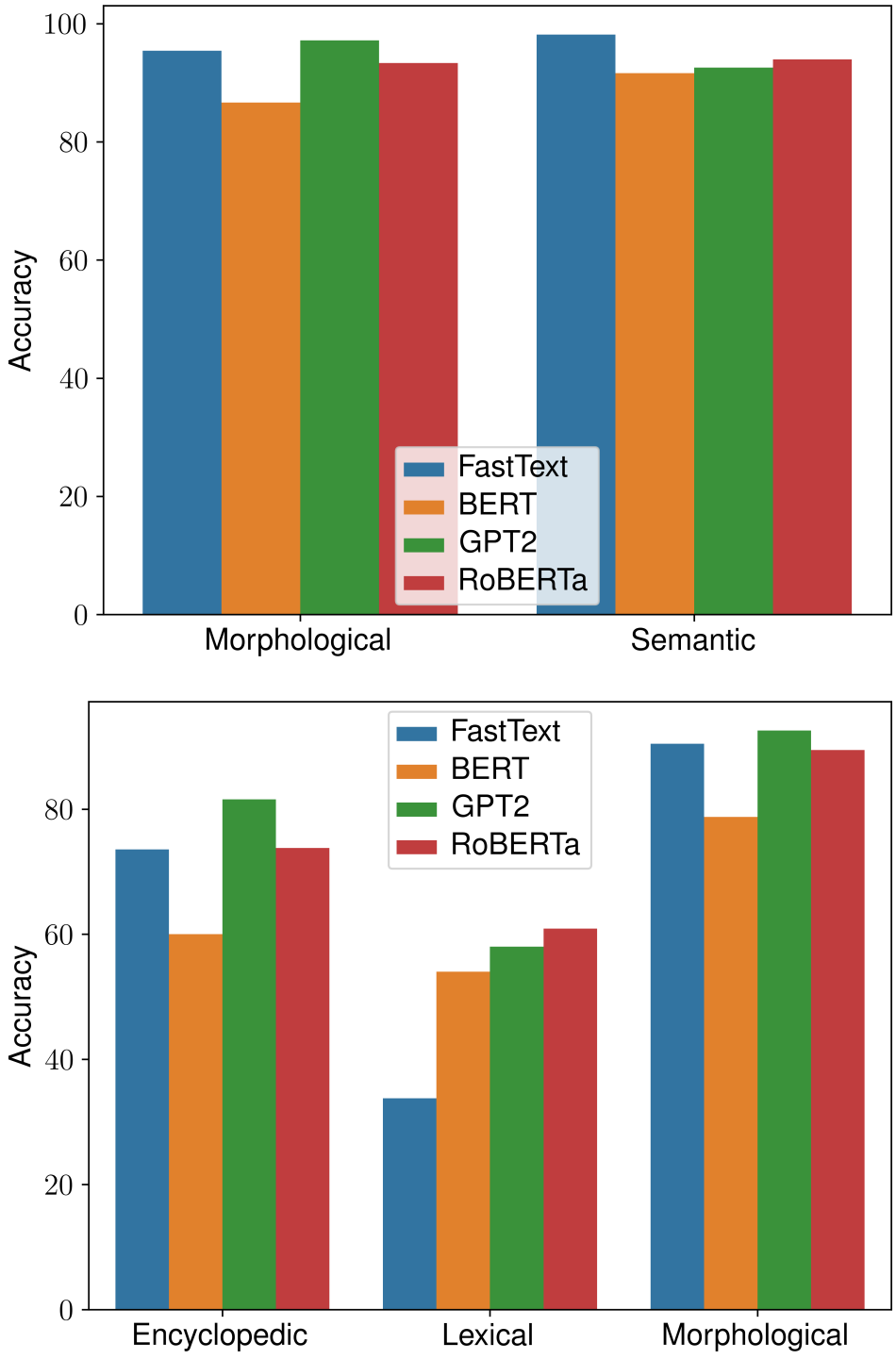

Figure 3.4: The test accuracy in Google (top) and BATS (bottom) results split by high-level categories.

Query	Candidates
hilarious:funny	right:wrong, hard:boring, nice:crazy, **great:good**
poor:money	**tired:energy**, angry:emotion, hot:ice, hungry:water
wrench:tool	cow:milk, radio:sound, tree:forest, **carrot:vegetable**
beautiful:pretty	**terrible:bad**, brave:valiant, new:old, tall:skinny
shield:protect	computer:talk, **vehicle:transport**, pencil:make, song:sing
sick:health	sad:emotion, tall:intelligence, **scared:courage**, smart:energy

Table 3.10: The model prediction examples from RoBERTa with s_{mPPL} tuned on the validation set. The gold answers are shown in bold, while the model predictions are underlined.

3.7.1 Error Analysis

Table 3.10 shows all examples from the U2 dataset of the easiest difficulty (i.e. grade 4), which were misclassified by RoBERTa, with s_{mPPL} tuned on the validation set. We can see a few typical issues with word embeddings and LMs. For instance, in the first example, the model confuses the antonym pair *right:wrong* with synonymy. In the second example, we have that someone who is poor lacks money, while someone who is hungry lacks food. However, the selected candidate pair is *hungy:water* rather than *hungry:food*, which is presumably chosen because water is assumed to be a near-synonym of food. In the third example (*wrench:tool*), the hypernymy relation is confused with a meronymy relation in the selected candidate *tree:forest*. In the last three examples, the model has selected answers which seem reasonable. In the fourth example, *beautiful:pretty*, *terrible:bad* and *brave:valiant* can all be considered to be synonym pairs. In the fifth example, *vehicle:transport* is clearly the correct answer, but the pair *song:sing* is nonetheless relationally similar to *shield:protect*. In the last example, we can think of being sad as an emotional state, like being sick is a health state, which provides some justification for the predicted answer. On the other hand, the gold answer is based on the argument that

	Mask	$\text{SAT}_{\text{SPLIT}}$	U2	U4	Google	BATS
BERT	full	41.8	44.7	41.2	88.8	67.9
	head	31.8	28.1	34.3	72.0	62.4
	tail	33.5	31.6	38.2	64.2	63.1
RoBERTa	full	53.4	58.3	57.4	93.6	78.4
	head	38.6	37.7	41.0	60.6	54.5
	tail	35.6	37.3	40.5	55.8	64.2

Table 3.11: The accuracy results by masking head or tail of the candidate answers. Results in the top row correspond to the full model without masking.

	Mask	Data	g_{pos}	t
BERT	head	$\text{SAT}_{\text{SPLIT}}$	val_5	*to-what*
		U2	val_5	*to-as*
		U4	mean	*to-as*
		Google	val_5	*she-as*
		BATS	val_5	*to-as*
	tail	$\text{SAT}_{\text{SPLIT}}$	val_3	*what-to*
		U2	val_7	*to-what*
		U4	val_4	*rel-same*
		Google	val_7	*as-what*
		BATS	val_7	*to-as*
RoBERTa	head	$\text{SAT}_{\text{SPLIT}}$	val_5	*as-what*
		U2	val_5	*rel-same*
		U4	val_7	*she-as*
		Google	val_5	*what-to*
		BATS	val_5	*she-as*
	tail	$\text{SAT}_{\text{SPLIT}}$	mean	*what-to*
		U2	val_7	*rel-same*
		U4	mean	*what-to*
		Google	val_7	*as-what*
		BATS	val_7	*what-to*

Table 3.12: The best configurations for the hypothesis-only scoring function.

someone who is sick lacks health like someone who is scared lacks courage.

3.7.2 Hypothesis Only Score

Recently, several researchers have found that standard NLP benchmarks, such as SNLI (Bowman et al., 2015) for language inference, contain several annotation artifacts that makes the task simpler for automatic models (Poliak et al., 2018; Gururangan et al., 2018). One of their most relevant findings is that models which do not even consider the premise can reach high accuracy. More generally, these issues have been found to be problematic in NLP models (Linzen, 2020) and neural networks more generally (Geirhos et al., 2020). According to the results shown in Table 3.6, we already found that the PMI baseline achieved a non-trivial performance, even outperforming BERT in a few settings and datasets. This suggests that several implausible negative examples are included in the analogy datasets. As a further exploration of such artifacts, here we analyse the analogue of a *hypothesis-only* baseline. In particular, for this analysis, we masked the head or tail of the candidate answer in all evaluation instances. Then, we test the masked LMs with the same AP configuration and tuning on these artificially-modified datasets. As can be seen in Table 3.11, a non-trivial performance is achieved for all datasets, which suggests that the words from the answer pair tend to be more similar to the words from the query than the words from negative examples. Table 3.12 includes the best configuration based on each validation set in for s_{PMI}, s_{mPPL} and the hypothesis-only baseline.

3.7.3 Additional Scoring Functions

As alternative scoring functions for LM, we have tried two other scores: PMI score based on masked token prediction (Davison et al., 2019b) (Mask PMI) and cosine

	Score	$\text{SAT}_{\text{SPLIT}}$	U2	U4	Google	BATS
BERT	embedding	24.0	22.4	26.6	28.2	28.3
	Mask PMI	25.2	23.3	31.5	61.2	46.2
	s_{PMI}	40.4	42.5	27.8	87.0	68.1
	s_{mPPL}	41.8	44.7	41.2	88.8	67.9
RoBERTa	embedding	40.4	42.5	27.8	87.0	68.1
	Mask PMI	43.0	36.8	39.4	69.2	58.3
	s_{PMI}	51.3	49.1	38.7	92.4	77.2
	s_{mPPL}	53.4	58.3	57.4	93.6	78.4

Table 3.13: The test accuracy tuned on each validation set.

similarity between the embedding difference of a relation pair similar to what used in word-embedding models. For embedding method, we give a prompted sentence to LM to get the last layer's hidden state for each word in the given pair and we take the difference between them, which we regard as the embedding vector for the pair. Finally we pick up the most similar candidate in terms of the cosine similarity with the query embedding. Table 3.13 shows the test accuracy on each dataset. As one can see, AP scores outperform other methods with a great margin.

3.7.4 Parameter Sensitivity

We found that optimal values of the parameters α and β are highly dependent on the dataset, while other parameters such as the template type t vary across LMs. On the other hand, as shown in Figure 3.5, the optimal permutations of the templates are relatively consistent, with the original ordering $a : b :: c : d$ typically achieving the best results. The results degrade most for permutations that mix the two word pairs (e.g. $a : c :: b : d$).

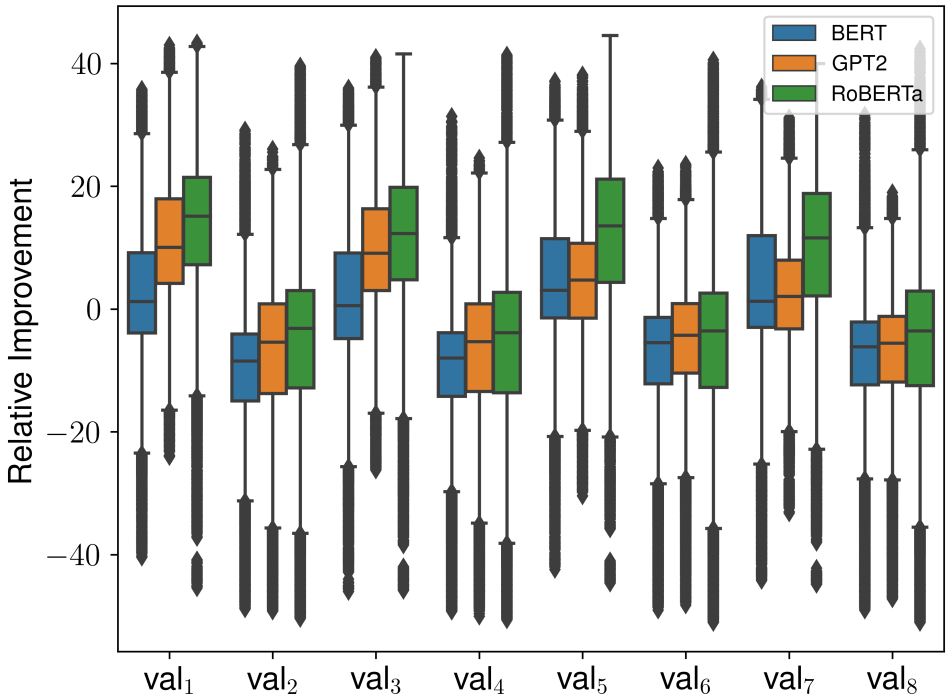

Figure 3.5: The box plot of the relative improvement on test accuracy in each dataset over all configurations of s_{mPPL} grouped by g_{pos}. Here val_k corresponds to kth positive permutation shown in Figure 3.2.

3.8 Summary

In this chapter, we have presented an extensive analysis of the ability of LMs to identify analogies, which is a unified set of datasets from various domain including educational settings, commonsense KG, named entities, and scientific metaphor. To this end, we proposed standard techniques to apply LMs to the unsupervised task of solving the analogy questions. First, we have shown the results of simple zero-shot approaches to solve the analogy questions with static word embeddings and LMs to establish the baselines. Then, we have introduced scoring functions tailored for solving analogy questions with LMs.

Our empirical results shed light on the strengths and limitations of various models. As a conclusion, LMs can identify analogies to a certain extent, but not all LMs are able to achieve a meaningful improvement over word embeddings (whose limitations in analogy tasks are well documented). On the other hand, when carefully tuned, some LMs are able to achieve SoTA results. We emphasize that results are highly sensitive to the chosen hyperparameters (which define the scoring function and the prompt among others). Finally, clearly LMs might still be able to learn to solve analogy tasks when given appropriate training data, so in next chapter, we will consider to learn the relation representation via LM fine-tuning for modelling word analogy as well as other relational knowledge-centric tasks.

4 Representing Relations with Language Models

4.1 Introduction

In the previous chapter, we have studied the methodologies of using LMs to solve analogy questions in unsupervised way (i.e., without training). We have shown that large LMs are capable of solving analogy questions to some extent, but they struggle to even outperform a static word embedding or a statistical method in the zero-shot setting. Furthermore, with a scoring function designed to evaluate AP tuned on the validation set, we have succeeded to improve some of the LMs such as RoBERTa to achieve SoTA result. According to the findings, it can be said that LMs have obtained relational knowledge at pre-training somewhat, but extracting such relational knowledge from LMs is non-trivial to be accomplished. Consequently, in this chapter, we focus on LM fine-tuning in order to learn the representation of lexical relation leveraging the relational knowledge of LMs.

In this chapter, we propose a framework to distill relational knowledge from a pre-trained LM to achieve a relation embedding model to represent the relationship be-

tween word pairs, which is different from any other aforementioned studies. Given a word pair and a underlying LM, we first create a sentence to represent the relationship between the word pair by a fixed template, and obtain contextualised embeddings of the sentence from the LM, that are then aggregated to get a fixed-length vector, which we refer as the relation embedding of the word pair from the LM. This relation embeddings are used to fine-tuned the LM with contrastive loss in a way that the relation embeddings in a similar relationship become closer, while further away from the embeddings of irrelevant relationship in the embedding space. Our main models, which we refer as *RelBERT*, are based on RoBERTa as the underlying LM, and fine-tuned on a modified version of a dataset from relational similarity of SemEval 2012 Task 2 (Jurgens et al., 2012), with Information Noise Constrastive Estimation (InfoNCE) (Oord et al., 2018). Despite the conceptual simplicity of this approach, the resulting model outperforms all the baselines including statistical approach (Turney, 2005), word embedding, and LMs including recent large scale models such as OPT and T5 in the zero-shot setting, meaning without any task-specific validation or additional model training. For example, RelBERT achieves 73% accuracy on SAT analogy question, that outperforms previous SoTA by 20 percentage points, and GPT-3 by 17 percentage points in the zero-shot setting. This is remarkable that LM can obtain surprisingly high-quality representations with a limited amount training dataset. Also, we find that RelBERT is surprisingly efficient as RelBERT based on $\text{RoBERTa}_{\text{BASE}}$, that has 100 million of parameters, already outperforms GPT-3, that has of 100 trillion of parameters in SAT analogy question. Overall, we test RelBERT in nine diverse analogy questions as well as five lexical relation classification datasets, and we show that RelBERT is capable of achieving the best result in all the analogy questions with great margin as well as being competitive as the SoTA in lexical relation classification.

To further understand the capability of RelBERT, we analyse RelBERT from various perspective in the analysis. One important finding is that RelBERT performs strongly even on relation types that are fundamentally different from the ones in the training data. For instance, while the training data involves standard lexical relations such as hypernymy, synonymy, meronyny and antonymy, the model is able to capture morphological relations, and even factual relations between named entities. Interestingly, RelBERT trained on the relation similarity dataset achieves better result in those unseen relation types than the model trained on them explicitly. Moreover, even if we remove all training examples for a given lexical relation (e.g. hypernymy), we find that the resulting model is still capable of modelling that relationship. These findings proves the generalization ability of RelBERT that enables to recognize word pairs with unseen relation types by extracting relation knowledge from the pre-trained LM, rather than merely generalising the examples from the training data. We find that, surprisingly, RelBERT achieves a non-trivial performance on named entities, despite only being trained on concepts. Moreover, on analogies between concepts, even the smallest RelBERT model, with 140M parameters, substantially outperforms all the considered LMs.

In this chapter, we first explain RelBERT, our framework for extracting relation embeddings from fine-tuned LMs, in § 4.2. Subsequently, § 4.5 compares the results of RelBERT with baselines, including a large number of recent large LMs. To understand the generalization ability of RelBERT, we analyze the performance of RelBERT in § 4.6.1, where we conduct an experiment in which certain relation type are excluded from the training set and evaluate the model on the excluded relation type. In addition to the main experiment, we compare RelBERT with conversational LMs in § 4.6.2.1 and few-shot learning in § 4.6.2.2 as well as different template to prompt LMs in § 4.6.2.3. As the learning process of RelBERT can be

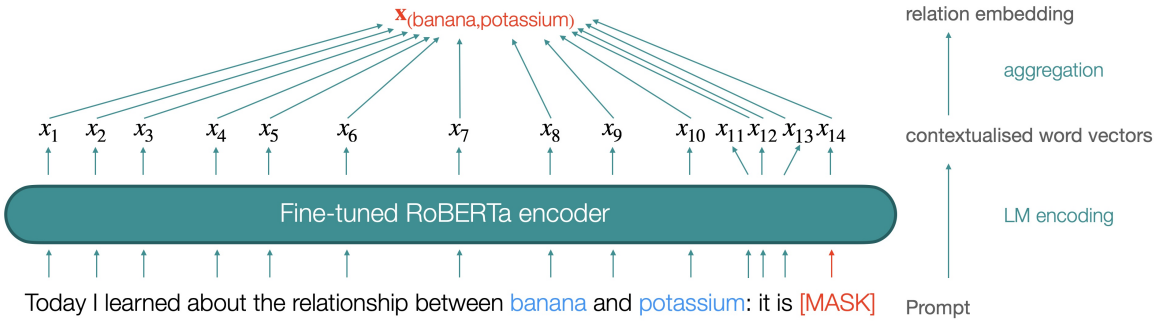

Figure 4.1: Schematic overview of the RelBERT model. A word pair is presented to an LM encoder using a prompt. A relation vector, capturing how the two input words are related, is then obtained by aggregating the contextualised embeddings from the output layer.

affected by many factors, we provide a comprehensive analysis of RelBERT fine-tuning in § 4.6.3, where we fine-tune RelBERT with different dataset in § 4.6.3.1, different loss functions in § 4.6.3.2, different batch size in § 4.6.3.3, different LMs in § 4.6.3.4, and different prompt templates in § 4.6.3.5. Finally, we provide a qualitative analysis of the relation embedding space of RelBERT § 4.6.4.

4.2 RelBERT

We now introduce our proposed RelBERT model, a fine-tuned LM encoder of the BERT family for modelling relational similarity. The input to RelBERT consists of a word pair, which is fed to the LM using a prompt. The LM itself is fine-tuned to map this input to a vector that encodes how the two given words are related. We will refer to this vector as a *relation embedding*. A schematic overview of the RelBERT model is shown in Figure 4.1. Our overall strategy is explained in more detail in § 4.2.1, while the details of the fine-tuning process are provided in § 4.2.2.

4.2.1 Overall Strategy

To obtain the relation embedding of a word pair (h, t), we need to construct a suitable input for the LM. While it is possible to simply use the pair (h, t) as input, similar to what is done by COMET (Bosselut et al., 2019), better results can be achieved by converting the word pair into a more or less naturally sounding sentence. This is true, in particular, because the amount of high-quality data that is available for training RelBERT is relatively limited, as seen in § 4.2.3. We thus need to manually create a template with placeholders for the two target words, which somehow expresses that we are interested in modelling the relationship between the two words. Such a strategy has already been proven effective for factual knowledge probing (Petroni et al., 2019a) and text classification (Schick and Schütze, 2021; Tam et al., 2021; Le Scao and Rush, 2021), among many others. Since we will rely on fine-tuning the LM, the exact formulation of the prompt matters less than in zero-shot settings. However, we found that performance suffers when the prompt is too short, in accordance with Bouraoui et al. (2020) and Jiang et al. (2020), or when the prompt is nonsensical (i.e. when it does not express the idea of modelling a relationship). With this in mind, we will use the following five templates for our main experiments:

1. Today, I finally discovered the relation between **[h]** and **[t]** : **[h]** is the `<mask>` of **[t]**
2. Today, I finally discovered the relation between **[h]** and **[t]** : **[t]** is **[h]**'s `<mask>`
3. Today, I finally discovered the relation between **[h]** and **[t]** : `<mask>`
4. I wasn't aware of this relationship, but I just read in the encyclopedia that

[**h**] is the `<mask>` of [**t**]

5. I wasn't aware of this relationship, but I just read in the encyclopedia that [**t**] is [**h**]'s `<mask>`

where `<mask>` is the LM's mask token, and [**h**] and [**t**] are slots that are filled with the head word h and tail word t from the given word pair. As a final step, we construct the relation embedding $\mathbf{x}_{(h,t)}$ from the contextualised representation of the prompt in the LM's output layer. In particular, we have experimented with the following three strategies:

- We take the contextualised representation of the `<mask>` token as the relation embedding (*average*).
- We average the contextualised embeddings across all tokens from the prompt (*mask*).
- We average the contextualised embeddings across all tokens from the prompt except for the `<mask>` token (*average w.o. mask*).

Next, we discuss how the model is trained.

4.2.2 Training Objective

The LM encoder used in RelBERT is initialised from a pre-trained RoBERTa model. It is then fine-tuned using a contrastive loss over the training dataset introduced in § 4.2.3, based on the idea that word pairs which belong to the same relation should have a relation embedding that is similar, whereas word pairs belonging to different

relations should have embeddings that are further apart. To this end, we assume access to a set of positive training examples $\mathcal{P}_r$ and a set of negative examples $\mathcal{N}_r$, for a number of relations $r \in \mathcal{R}$. In particular, $\mathcal{P}_r$ contains word pairs (h, t) which belong to relation r, whereas $\mathcal{N}_r$ contains examples of word pairs which do not. We consider three different loss functions to implement the proposed idea.

4.2.2.1 Triplet Loss

The *triplet loss* (Schroff et al., 2015) relies on training data in the form of triples (a, p, n), where a is called the anchor, p is a positive example. and n is a negative example. The aim of this loss is to ensure that the distance between a and p is smaller, by some margin, than the distance between a and n. In our case, the elements a, p and n correspond to word pairs, where $a, p \in \mathcal{P}_r$ and $n \in \mathcal{N}_r$ for some relation r. Let us write $\boldsymbol{x}_a$ for the relation embedding of a word pair a. We then have the following loss:

$$L_{\text{tri}} = \sum_{r \in \mathcal{R}} \sum_{(a,p,n) \in \mathcal{P}_r \times \mathcal{P}_r \times \mathcal{N}_r} \max\left(0, \mid \boldsymbol{x}_a - \boldsymbol{x}_p \mid - \mid \boldsymbol{x}_a - \boldsymbol{x}_n \mid + \Delta\right) \tag{4.1}$$

where $\Delta > 0$ is the margin and $\mid \cdot \mid$ is the l^2 norm.

4.2.2.2 InfoNCE

InfoNCE (Oord et al., 2018) addresses two potential limitations of the triplet loss. First, while the triplet loss only considers one negative example at a time, InfoNCE can efficiently contrast each positive example to a whole batch of negative examples. Second, while the triplet loss uses the l^2 norm, InfoNCE relies on the cosine

similarity, which tends to be better suited for comparing embeddings. The InfoNCE loss can be defined as follows:

$$L_{\text{nce}} = \sum_{r \in \mathcal{R}} \sum_{(a,p) \in \mathcal{P}_r \times \mathcal{P}_r} \left(-\log \frac{\exp\left(\frac{\cos(\boldsymbol{x}_a, \boldsymbol{x}_p)}{\tau}\right)}{\exp\left(\frac{\cos(\boldsymbol{x}_a, \boldsymbol{x}_p)}{\tau}\right) + \sum_{n \in \mathcal{N}_r} \exp\left(\frac{\cos(\boldsymbol{x}_a, \boldsymbol{x}_n)}{\tau}\right)} \right) \tag{4.2}$$

where τ is a temperature parameter to control the scale of the exponential and cos is the cosine similarity.

4.2.2.3 InfoLOOB

Info-Leave-One-Out-Bound (InfoLOOB) (Fürst et al., 2021) is a variant of InfoNCE, in which the positive example is omitted from the denominator. This is aimed at preventing the saturation of the loss value, which can occur with InfoNCE due to dominant positives. Applied to our setting, the loss is as follows:

$$L_{\text{loob}} = \sum_{r \in \mathcal{R}} \sum_{(a,p) \in \mathcal{P}_r \times \mathcal{P}_r} \left(-\log \frac{\exp\left(\frac{\cos(\boldsymbol{x}_a, \boldsymbol{x}_p)}{\tau}\right)}{\sum_{n \in \mathcal{N}_r} \exp\left(\frac{\cos(\boldsymbol{x}_a, \boldsymbol{x}_n)}{\tau}\right)} \right) \tag{4.3}$$

4.2.3 Training Data

As described in § 4.2.2, to train RelBERT we need positive and negative examples of word pairs belonging to particular relations. In this section, we describe the four datasets that we considered for training RelBERT. The main properties of these

Relation	Examples
Case Relation	['designer', 'fashions'], ['preacher', 'parishioner'], ['hunter', 'rifle']
Meronym (Part-Whole)	['building', 'wall'], ['team', 'player'], ['movie', 'scene']
Antonym (Contrast)	['smooth', 'rough'], ['difficult', 'easy'], ['birth', 'death']
Space-Time	['refrigerator', 'food'], ['factory', 'product'], ['pool', 'swimming']
Representation	['diploma', 'education'], ['groan', 'pain'], ['king', 'crown']
Hypernym (Class Inclusion)	['furniture', 'chair'], ['furniture', 'chair'], ['flower', 'daisy']
Synonym (Similar)	['couch', 'sofa'], ['sadness', 'melancholia'], ['confident', 'arrogance']
Attribute	['steel', 'strong'], ['glass', 'shattered'], ['miser', 'greed']
Non Attribute	['empty', 'full'], ['incomprehensible', 'understood'], ['destitution', 'abundance']
Cause-Purpose	['tragedy', 'tears'], ['fright', 'scream'], ['battery', 'laptop']

Table 4.1: Examples of word pairs from each parent relation category in the RelSim dataset.

datasets are summarised in Table 4.2. We now present each dataset in more detail. For each dataset, we have a training and validation spit, which are used for training RelBERT and for selecting the hyperparameters.

4.2.3.1 RelSim

The RelSim[1] was introduced for SemEval 2012 Task 2 (Jurgens et al., 2012). It contains crowdsourced judgements about 79 fine-grained semantic relations, which

[1]Our preprocessed version of this dataset is available at https://huggingface.co/datasets/relbert/semeval2012_relational_similarity; the original dataset is available at https://sites.google.com/site/semeval2012task2/download.

Dataset	Relational Similarity Dataset (RelSim)	ConceptNet
#relations	89/89/-	28/18/16
Average #positive examples per relation	14.7/3.7/-	20,824/66/74
Relation Hierarchy	True	False
Domain	Concepts	Concepts
Dataset	NELL	T-REX
#relations	31/4/6	721/602/24
Average #positive examples per relation	177/219/225	1,767/529/4
Relation Hierarchy	False	False
Domain	Named entities	Named entities

Table 4.2: Statistics of the training sets that are considered for RelBERT, including the number of relations, the average number of triples per relation in the training / validation / test sets, and the number of unique positive triples; we also specify whether the relations are organised in a hierarchy, the domain from which the entities are coming.

are grouped into 10 parent categories. Table 4.1 shows word pairs randomly sampled from the highest ranked word pairs in each parent category of RelSim. For each semantic relation, a list of word pairs is provided in RelSim, with each word pair being assigned a prototypicality score, where we used the platinum ratings from the original dataset. To convert this dataset into the format that we need for training RelBERT, we consider the 79 fine-grained relations and the 10 parent relations separately. For the fine-grained relations, we choose the 10 most prototypical word pairs, i.e. the word pairs with the highest scores, as positive examples, while the 10 lowest ranked word pairs are used as negative examples. For the parent relations, the set of positive examples contains the positive word pairs of each of the fine-grained relations that belong to the parent relation. The negative examples for the parent relations are taken to be the positive examples of the other relations. This is because the parent relations are mutually exclusive, whereas the semantic distinction between the fine-grained relations is often very subtle. From the resulting dataset, we randomly choose 80% of the word pairs for training, and we keep

the remaining 20% as a validation set.

4.2.3.2 ConceptNet

ConceptNet[2] (Speer and Havasi, 2012) is a commonsense knowledge graph. It encodes semantic relations between concepts, which can be single nouns or short phrases. The knowledge graphs refers to a total of 34 different relations. Since the original ConceptNet contains more than two millions of triples, we employ the version released from Li et al. (2016), where the triples are filtered by their confidence score. We use the *test set* consisting of the 1200 most confident tuples as an evaluation dataset, the *dev1* and *dev2* sets consisting of the next 1200 most confident tuples as our validation set, and the *training set* consisting of 600k tuples as our training set[3]. We have disregarded any triples with negated relations such as *NotCapableOf* or *NotDesires*, because they essentially indicate the lack of a relationship. The positive examples for a given relation are simply the word pairs which are asserted to have this relation in the knowledge graph. The negative examples for a given relation are taken to be the positive examples for the other relations, i.e. $\mathcal{N}_r = \{(a, b) \in \mathcal{P}_{\hat{r}} | \hat{r} \in \mathcal{R} \setminus \{r\}\}$.

[2]Our preprocesed version of this dataset is available at https://huggingface.co/datasets/relbert/conceptnet_relational_similarity; the original dataset is available at https://conceptnet.io/.

[3]The filtered version of ConceptNet is available at https://home.ttic.edu/~kgimpel/commonsense.html.

4.2.3.3 NELL-One

NELL[4] (Mitchell et al., 2018) is a system to collect structured knowledge from web. The authors of Xiong et al. (2018) compiled and cleaned up the latest dump file of NELL at the time of publication to create a knowledge graph, called NELL-One for one-shot relational learning. We employ NELL-One with its original split from Xiong et al. (2018), which avoids any overlap between the relation types appearing in the test set, on the one hand, and the relation types appearing in the training and validation sets, on the other hand. Similar as for ConceptNet, the positive examples for a given relation are the word pairs that are asserted to belong to that relation in the training set, whereas the negative examples for a relation are the positive examples of the other relations in the training set.

4.2.3.4 T-REX

T-REX[5] (Elsahar et al., 2018) is a knowledge base that was constructed by aligning Wikipedia and Wikidata. It contains a total of 20 million triples, all of which are aligned with sentences from introductory sections of Wikipedia articles. We first remove triples if either their head or tail is not a named entity, which reduces the number of triples from 20,877,472 to 12,561,573, and the number of relations from 1,616 to 1,470. Then, we remove relations with fewer than three triples, as we need at least three triples for each relation type, which reduces the number of

[4]Our preprocessed version of this dataset is available at https://huggingface.co/datasets/relbert/nell_relational_similarity; the original dataset is available at https://github.com/xwhan/One-shot-Relational-Learning.

[5]Our preprocessed version of this dataset is available at https://huggingface.co/datasets/relbert/t_rex_relational_similarity; the original dataset is available at https://hadyelsahar.github.io/t-rex/.

triples to 12,561,250, while the number of relations to 1,237. One problem with this dataset is that it contains a number of distinct relations which intuitively have the same meaning. For example, the relations *band* and *music by* both represent "A song played by a musician". Therefore, we manually mapped such relations onto the same type. Also, we manually removed overly vague relations. For example, the relationship *is a* refers to "hypernym of", but T-REX contains other relation types of more specific hypenym relation types such as *fruit of*, *religion*, and *genre*, and *is a* contains triples from those relation types. Another example is *is in*, which contains triples with the relation of "located in", but T-REX contains fine-grained relationships of "located in" such as *town*, *state*, *home field*, and *railway line*, and *is in* contains triples from those relation types.

4.3 Evaluation Tasks

We evaluate RelBERT on two relation-centric tasks: analogy questions (unsupervised), and lexical relation classification (supervised). In this section, we describe these tasks and introduce the benchmarks we considered.

4.3.1 Analogy Question

We extend the five analogy question datasets introduced in § 3.2, by adding four new datasets. We first converted the validation and test splits of T-REX, NELL and ConceptNet, which were introduced in § 4.2.3, into the format of analogy questions. Note that the validation split is used in our ablation study to compare RelBERT training set, but not used in the main experiment where we solve analogy

Dataset	Avg. #Answer Candidates	#Questions
SAT	5	- / 374
U2	4	24 / 228
U4	4	48 / 432
Google	4	50 / 500
BATS	4	199 / 1,799
SCAN	74	178 / 1,616
NELL-One	6	400 / 600
T-REX	67	496 / 183
ConceptNet	18	1,112 / 1,192

Table 4.3: Main statistics of the analogy question datasets, showing the average number of answer candidates, and the total number of questions (validation / test).

question in the zero-shot setting. Thus, we do not consider approaches that requires validation as well as training such as § 3.4. These analogy questions were constructed by taking two word pairs from the same relation type, one of which is used as the query while the other is used as the correct answer. For the query and its correct word pair from a relation type, we consider all the word pairs from other relation types as its negative candidates. Finally, we include a new analogy question converted from SCAN (Czinczoll et al., 2022), that is a dataset for the relation mapping problem (Turney, 2008). The relation mapping problem is to find a bijective mapping between a set of relations from some source domain and a corresponding set of relations from a given target domain. SCAN[6] (Czinczoll et al., 2022) is an extension. Where Turney (2008) contains 10 scientific and 10 metaphorical domains, SCAN extends them by another 443 metaphorical domains and 2 scientific domains. A single SCAN instance contains a list of the source and the target words ($\boldsymbol{a} = [a_1, \ldots, a_m]$ and $\boldsymbol{b} = [b_1, \ldots, b_m]$). We convert such an instance into an analogy question, where the query is $[a_i, a_j]$ and the ground truth is $[b_i, b_j]$ and the negative candidates are $[b_{\hat{i}}, b_{\hat{j}}]$ for $\{(\hat{i}, \hat{j}) \in \{1, \ldots, m\} \times \{1, \ldots, m\} | (\hat{i}, \hat{j}) \neq (i, j)\}$.

[6]The original dataset is available at https://github.com/taczin/SCAN_analogies.

Dataset	Domain	Example
SAT	College Admission Test	[beauty, aesthete, pleasure, hedonist]
U2	Grade4	[rock, hard, water, wet]
	Grade5	[hurricane, storm, table, furniture]
	Grade6	[microwave, heat, refrigerator, cool]
	Grade7	[clumsy, grace, doubtful, faith]
	Grade8	[hidden, visible, flimsy, sturdy]
	Grade9	[panacea, cure, contagion, infect]
	Grade10	[grain, silo, water, reservoir]
	Grade11	[thwart, frustrate, laud, praise]
	Grade12	[lie, prevaricate, waver, falter]
U4	Low Intermediate	[accident, unintended, villain, evil]
	Low Advanced	[galleon, sail, quarantine, isolate]
	High Beginning	[salesman, sell, mechanic, repair]
	High Intermediate	[classroom, desk, church, pew]
	High Advanced	[erudite, uneducated, fervid, dispassionate]
BATS	Inflectional Morphology	[neat, neater, tasty, tastier]
	Derivational Morphology	[available, unavailable, interrupted, uninterrupted]
	Encyclopedic Semantics	[stockholm, sweden, belgrade, serbia]
	Lexicographic Semantics	[elephant, herd, flower, bouquet]
Google	Encyclopedic Semantics	[Canada, dollar, Croatia, kuna]
	Morphological	[happy, happily, immediate, immediately]
SCAN	Metaphor	[grounds for a building, solid, reasons for a theory, rational]
	Science	[conformance, breeding, adaptation, mating]
NELL	Named Entities	[Miami Dolphins, Cam Cameron, Georgia Tech, Paul Johnson]
T-REX	Named Entities	[Washington, Federalist Party, Nelson Mandela, ANC]
ConceptNet	Concepts	[bottle, plastic, book, paper]

Table 4.4: An example from each domain of the analogy question benchmarks.

This results in 178 and 1,616 questions for the validation and test sets, respectively. The number of answer candidates per question is 74 on average, which makes this benchmark particularly challenging. Table 4.3 summarises the main features of the analogy question datasets, and Table 4.4 shows an example from each of the anal-

	BLESS	CogALex	EVALution
Antonym	-	241 / 360	1095 / 90 / 415
Attribute	1892 / 143 / 696	-	903 / 72 / 322
Co-hyponym	2529 / 154 / 882	-	-
Event	2657 / 212 / 955	-	-
Hypernym	924 / 63 / 350	255 / 382	1327 / 94 / 459
Meronym	2051 / 146 / 746	163 / 224	218 / 13 / 86
Possession	-	-	377 / 25 / 142
Random	8529 / 609 / 3008	2228 / 3059	-
Synonym	-	167 / 235	759 / 50 / 277

	K&H+N	ROOT09
Antonym	-	-
Attribute	-	-
Co-hyponym	18134 / 1313 / 6349	2222 / 162 / 816
Event	-	-
Hypernym	3048 / 202 / 1042	2232 / 149 / 809
Meronym	755 / 48 / 240	-
Possession	-	-
Random	18319 / 1313 / 6746	4479 / 327 / 1566
Synonym	-	-

Table 4.5: Number of instances for each relation type across training / validation / test sets of all lexical relation classification datasets.

ogy questions.

4.3.2 Lexical Relation Classification

We consider the supervised task of relation classification. This task amounts to classifying word pairs into a predefined set of possible relation types[7]. To solve this task, we train a multi-layer perceptron with one hidden layer, which takes the RelBERT relation embedding of the word pair as input. The RelBERT encoder itself is frozen, since our focus is on evaluating the quality of the RelBERT relation embeddings. We consider the following widely-used multi-class relation classification

[7]Preprocessed versions of the datasets are available at https://huggingface.co/datasets/relbert/lexical_relation_classification.

	LM	Template	Epoch
$\text{RelBERT}_{\text{BASE}}$	$\text{RoBERTa}_{\text{BASE}}$	1	8
$\text{RelBERT}_{\text{LARGE}}$	$\text{RoBERTa}_{\text{LARGE}}$	4	9

Table 4.6: The best configuration of the template and the number of epoch for the main RelBERT models.

benchmarks: K&H+N (Necşulescu et al., 2015), BLESS (Baroni and Lenci, 2011), ROOT09 (Santus et al., 2016b), EVALution (Santus et al., 2015), and CogALex-V Subtask 2 (Santus et al., 2016a). Table 4.5 shows the size of the training, validation and test splits for each of these datasets, as well as the kinds of relations they cover. The hyperparameters of the multi-layer perceptron classifier are tuned on the validation split of each dataset. In particular, we tune the learning rate from $[0.001, 0.0001, 0.00001]$ and the hidden layer size from $[100, 150, 200]$. CogALex-V has no validation split, so for this dataset we employ the default configuration of Scikit-Learn (Pedregosa et al., 2011a), which uses a 100-dimensional hidden layer and is optimized using Adam with a learning rate of 0.001.

4.4 Experimental Settings

In section, we explain the RelBERT training details § 4.4.1, and we introduce the baselines for analogy questions in § 4.4.2, and lexical relation classification in § 4.4.3.

4.4.1 RelBERT Training

In our experiments, we consider a number of variants of RelBERT, which differ in terms of the pre-trained LM that was used for initialising the model, the loss func-

tion § 4.2.2, and the training data § 4.2.3. In each case, RelBERT is trained for 10 epochs. Moreover, we train one RelBERT model for each of the five prompt templates § 4.2.1. The final model is obtained by selecting the epoch and prompt template that achieved the best performance on the validation split, in terms of accuracy. See § 4.6.5 to find the best hyperparameter used in each experiment. The default configuration for RelBERT is to fine-tune a $\text{RoBERTa}_{\text{BASE}}$[8] or $\text{RoBERTa}_{\text{LARGE}}$[9] model using InfoNCE on the RelSim dataset. We will refer to the resulting models as $\text{RelBERT}_{\text{BASE}}$[10] and $\text{RelBERT}_{\text{LARGE}}$[11] respectively. The other hyper-parameters are fixed as follows. When using the triplet loss, we set the margin Δ to 1, the learning rate to 0.00002 and the batch size to 32. When using InfoNCE or InfoLOOB, we set the temperature τ to 0.5, the learning rate to 0.000005 and the batch size to 400. In all cases, we fix the random seed as 0 and we use Adam (Kingma and Ba, 2014) as the optimiser. To select the the aggregation strategy, as a preliminary experiment, we fine-tuned $\text{RelBERT}_{\text{BASE}}$ with each of the three strategies suggested in § 4.2.2. As we found that *average w.o. mask* achieved the best accuracy on the validation set of RelSim, we used this as the default aggregation strategy. Table 4.6 shows the configuration of RelBERT models we obtained.

4.4.2 Baselines for Analogy Question

As the baselines for analogy question, we consider the zero-shot baselines we described in § 3.3. As a static word embedding baseline, we use fastText, and as LM baselines, we consider BERT and RoBERTa as MLMs, GPT-2, GPT-J (Wang and

[8] https://huggingface.co/roberta-base
[9] https://huggingface.co/roberta-large
[10] https://huggingface.co/relbert/relbert-roberta-base
[11] https://huggingface.co/relbert/relbert-roberta-large

	Model	Inst-FT	Model Size	Name on HuggingFace
MLM	$\text{BERT}_{\text{BASE}}$		110M	`bert-base-cased`
	$\text{BERT}_{\text{LARGE}}$		355M	`bert-large-cased`
	$\text{RoBERTa}_{\text{BASE}}$		110M	`roberta-base`
	$\text{RoBERTa}_{\text{LARGE}}$		355M	`roberta-large`
CLM	$\text{GPT-2}_{\text{SMALL}}$		124M	`gpt2`
	$\text{GPT-2}_{\text{BASE}}$		355M	`gpt2-medium`
	$\text{GPT-2}_{\text{LARGE}}$		774M	`gpt2-large`
	GPT-2_{XL}		1.5B	`gpt2-xl`
	$\text{GPT-J}_{\text{125M}}$		125M	`EleutherAI/gpt-neo-125M`
	$\text{GPT-J}_{\text{1.3B}}$		1.3B	`EleutherAI/gpt-neo-1.3B`
	$\text{GPT-J}_{\text{2.7B}}$		2.7B	`EleutherAI/gpt-neo-2.7B`
	GPT-J_{6B}		6B	`EleutherAI/gpt-j-6B`
	$\text{GPT-J}_{\text{20B}}$		20B	`EleutherAI/gpt-neox-20b`
	OPT_{125M}		125M	`facebook/opt-125m`
	OPT_{350M}		350M	`facebook/opt-350m`
	$\text{OPT}_{\text{1.3B}}$		1.3B	`facebook/opt-1.3b`
	OPT_{30B}		30B	`facebook/opt-30b`
	$\text{OPT-IML}_{\text{1.3B}}$	✓	1.3B	`facebook/opt-iml-1.3b`
	$\text{OPT-IML}_{\text{30B}}$	✓	30B	`facebook/opt-iml-30b`
	$\text{OPT-IML}_{\text{M-1.3B}}$	✓	1.3B	`facebook/opt-iml-max-1.3b`
	$\text{OPT-IML}_{\text{M-30B}}$	✓	30B	`facebook/opt-iml-max-30b`
EDLM	T5_{SMALL}		60M	`t5-small`
	T5_{BASE}		220M	`t5-base`
	T5_{LARGE}		770M	`t5-large`
	T5_{3B}		3B	`t5-3b`
	T5_{11B}		11B	`t5-11b`
	$\text{Flan-T5}_{\text{SMALL}}$	✓	60M	`google/flan-t5-small`
	$\text{Flan-T5}_{\text{BASE}}$	✓	220M	`google/flan-t5-base`
	$\text{Flan-T5}_{\text{LARGE}}$	✓	770M	`google/flan-t5-large`
	$\text{Flan-T5}_{\text{XL}}$	✓	3B	`google/flan-t5-xl`
	$\text{Flan-T5}_{\text{XXL}}$	✓	11B	`google/flan-t5-xxl`
	Flan-UL2	✓	20B	`google/flan-ul2`

Table 4.7: The model checkpoints used in the LM baselines on HuggingFace model hub. All the model can be obtained at `https://huggingface.co`.

Komatsuzaki, 2021), OPT (Zhang et al., 2022), OPT-IML (Iyer et al., 2022) as CLMs, and T5 (Raffel et al., 2020), Flan-T5 (Chung et al., 2022), Flan-UL2 (Tay et al., 2023) as EDLMs. We rely on the weights that were shared by HuggingFace

for all pre-trained LMs. A complete list of the models we used can be found in Table 4.7.

Additionally, we consider GPT-3 released from OpenAI[12] as a commercial API, that is one of the largest private in-house LM at the moment. Note that the models on OpenAI API are subject to be changed every six months, so we cannot ensure the reproducibility of the result. We use the latest endpoint at the time being (May 2023) of `davinci`, the best GPT-3 model, and follow the same approach as the other public LMs explained in § 3.3.4, where we use the perplexity with the template of the analogical statement to choose the answer. We also consider the baseline on SAT, taken from the original paper of GPT-3 (Brown et al., 2020), which we refer as GPT-3_{original}.

4.4.3 Baselines for Lexical Relation Classification

LexNet (Shwartz and Dagan, 2016) and SphereRE (Wang et al., 2019a) are the current SoTA classifiers on the considered lexical relation classification datasets. Both methods rely on static word embeddings (Mikolov et al., 2013a; Bojanowski et al., 2016). LexNet trains an LSTM on the word pair by considering it as a sequence of two words, where each word is mapped to its feature map consisting of a number of lexical features such as part-of-speech and the word embedding. SphereRE employs hyperspherical learning (Liu et al., 2017) on top of the word embeddings, which is to learn a feature map from word embeddings of the word pairs to their relation embeddings, which are distributed over the hyperspherical space. In addition to those SoTA methods, we use a simple baseline based on word embeddings.

[12] https://openai.com/

Specifically, we train an multi-layer perceptron with a hidden layer in the same way as explained in § 4.3.2. As possible input representations for this classifier, we consider the concatenation of the word embeddings (*cat*) and the vector difference of the word embeddings (*diff*), possibly augmented with the component-wise product of the word embeddings (*cat+dot* and *diff+dot*). We experiment with word embeddings from GloVE[13], (Pennington et al., 2014) and fastText, where we use the same embedding model used in § 3.3.3.

4.5 Experimental Results

In this section, we report the experimental results for the analogy questions benchmarks in § 4.5.1 and for lexical relation classification in § 4.5.3.

4.5.1 Results on Analogy Questions

Table 4.8 shows the results for each analogy question benchmark in terms of accuracy. We can see that RelBERT substantially outperforms the baselines in all cases, where $\text{RelBERT}_{\text{BASE}}$ is the best for T-REX, and $\text{RelBERT}_{\text{LARGE}}$ is the best for the remaining datasets. Figure 4.2 plots the accuracy of each LM in function of model size along with the RelBERT, and we can see that the RelBERT models achieve the best result despite being two orders of magnitude smaller than $\text{Flan-T5}_{\text{XXL}}$ and Flan-UL2. Interestingly, RoBERTa usually outperforms the other LM baselines of comparable size, except on NELL and SCAN. This suggests that the strong performance of RelBERT is at least in part due to the use of RoBERTa as the underlying

[13]The embedding model is available from https://nlp.stanford.edu/projects/glove/.

	Model	SAT	U2	U4	Google	BATS	SCAN	NELL	T-REX	ConceptNet	Average
	Random	20.0	23.6	24.2	25.0	25.0	2.5	14.3	2.1	5.9	15.8
	LRA	*56.4*	-	-	-	-	-	-	-	-	-
	fastText	47.1	38.2	38.4	94.6	70.7	21.7	59.8	23.0	15.2	45.1
MLM	$BERT_{BASE}$	34.5	35.1	35.2	67.6	48.4	14.5	25.2	9.8	9.4	31.1
MLM	$BERT_{LARGE}$	32.9	36.0	36.6	79.8	59.4	14.1	32.0	14.2	14.0	35.4
MLM	$RoBERTa_{BASE}$	36.4	42.1	43.3	81.0	61.8	10.6	29.0	14.8	16.6	37.3
MLM	$RoBERTa_{LARGE}$	40.6	50.4	49.8	88.8	69.9	12.1	38.2	36.1	16.7	44.7
CLM	$GPT\text{-}2_{SMALL}$	32.1	38.2	36.8	56.8	43.6	5.1	28.5	7.1	8.1	28.5
CLM	$GPT\text{-}2_{BASE}$	34.8	42.5	40.5	75.8	58.3	7.0	43.8	6.6	11.6	35.7
CLM	$GPT\text{-}2_{LARGE}$	36.1	42.1	42.6	75.4	60.1	8.8	39.0	12.6	12.8	36.6
CLM	$GPT\text{-}2_{XL}$	36.9	43.0	44.0	80.4	62.0	8.2	40.2	11.5	12.6	37.6
CLM	$GPT\text{-}J_{125M}$	34.0	36.8	35.6	52.8	46.6	5.6	31.8	10.9	8.4	29.2
CLM	$GPT\text{-}J_{1.3B}$	36.6	42.1	42.1	77.0	63.1	8.8	47.3	13.1	11.7	38.0
CLM	$GPT\text{-}J_{2.7B}$	38.5	44.7	43.5	83.0	63.8	8.8	40.0	16.9	14.1	39.3
CLM	$GPT\text{-}J_{6B}$	45.5	48.7	47.0	87.4	67.9	9.5	43.8	20.2	14.0	42.7
CLM	$GPT\text{-}J_{20B}$	42.8	47.8	53.7	86.4	71.3	10.0	37.2	31.7	15.7	44.1
CLM	$GPT\text{-}3_{original}$*	*53.7*	-	-	-	-	-	-	-	-	-
CLM	$GPT\text{-}3_{davinci}$*	51.8	53.5	53.2	86.0	70.8	0.84	37.8	20.7	14.2	43.9
CLM	OPT_{125M}	33.7	37.3	35.4	60.0	46.1	7.1	39.7	10.4	10.8	31.2
CLM	OPT_{350M}	34.0	36.8	39.1	74.2	54.8	8.2	44.7	10.9	10.6	34.8
CLM	$OPT_{1.3B}$	38.5	40.4	43.5	83.4	62.8	8.8	46.3	11.5	14.8	38.9
CLM	OPT_{30B}	47.1	52.2	51.9	88.2	71.5	10.5	45.2	20.8	19.1	45.2
CLM	$OPT\text{-}IML_{1.3B}$	41.4	42.5	44.9	82.4	63.1	8.1	42.0	7.7	14.8	38.5
CLM	$OPT\text{-}IML_{30B}$	48.9	50.2	49.0	88.3	70.8	10.8	44.4	23.2	17.5	44.8
CLM	$OPT\text{-}IML_{M\text{-}1.3B}$	40.6	40.8	44.0	85.4	64.1	9.0	45.7	8.2	15.9	39.3
CLM	$OPT\text{-}IML_{M\text{-}30B}$	48.9	50.2	49.0	88.3	70.8	10.8	44.4	23.2	17.5	44.8
EDLM	$T5_{SMALL}$	28.9	32.9	30.6	55.4	41.0	17.0	38.0	15.8	5.8	29.5
EDLM	$T5_{BASE}$	26.7	34.6	39.8	41.4	43.0	9.9	27.8	6.0	7.7	26.3
EDLM	$T5_{LARGE}$	30.2	35.5	40.0	51.4	48.9	13.7	25.2	11.5	9.1	29.5
EDLM	$T5_{3B}$	34.8	35.1	35.4	45.8	41.6	10.2	25.3	20.8	9.0	28.7
EDLM	$T5_{11B}$	35.6	43.0	47.5	74.6	59.9	17.9	40.8	27.3	13.8	40.0
EDLM	$Flan\text{-}T5_{SMALL}$	26.7	36.4	41.7	49.0	42.7	11.6	35.5	16.9	8.3	29.9
EDLM	$Flan\text{-}T5_{BASE}$	31.8	38.6	42.1	63.6	51.4	13.7	38.8	20.8	9.6	34.5
EDLM	$Flan\text{-}T5_{LARGE}$	36.1	40.8	43.5	63.8	52.6	13.4	38.0	20.8	11.3	35.6
EDLM	$Flan\text{-}T5_{XL}$	42.0	49.6	50.7	86.8	66.8	17.6	46.0	30.6	17.0	45.2
EDLM	$Flan\text{-}T5_{XXL}$	52.4	55.7	55.6	91.2	74.7	16.3	43.8	26.8	17.9	48.3
EDLM	Flan-UL2	50.0	53.1	57.2	91.8	74.9	13.4	53.8	36.1	16.8	49.7
	$RelBERT_{BASE}$	59.9	59.6	57.4	89.2	70.3	25.9	62.0	**66.7**	39.8	59.0
	$RelBERT_{LARGE}$	**73.3**	**67.5**	**63.0**	**95.2**	**80.9**	**27.2**	**65.8**	64.5	**47.5**	**65.0**

Table 4.8: The accuracy on each analogy question dataset and the averaged accuracy across datasets, where the best model in each dataset shown in bold. Result in italics were taken from the original paper.

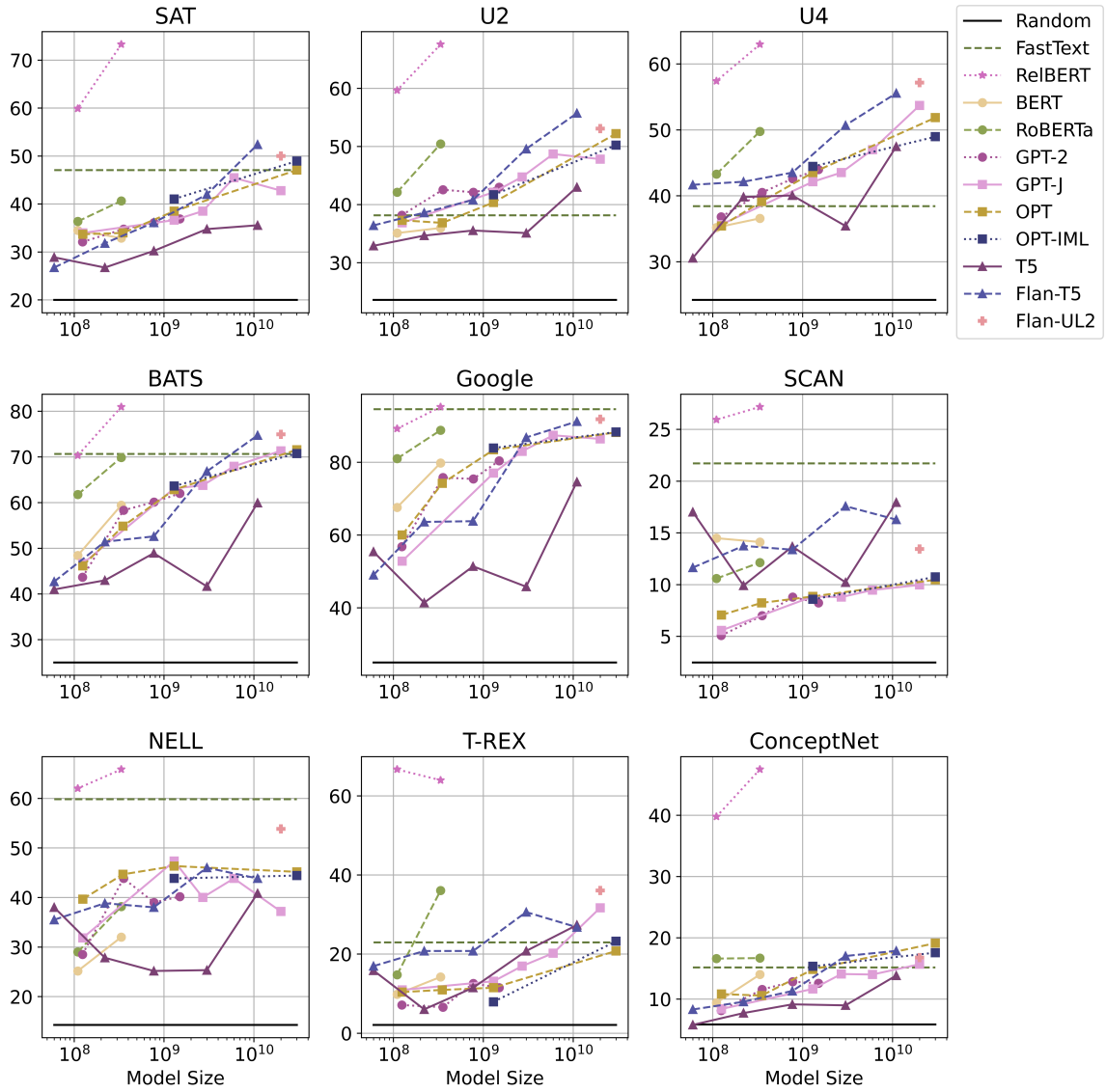

Figure 4.2: Accuracy on each analogy question dataset in function of the number of parameters in each LM.

model. Our analysis in § 4.6.3.4 will provide further support for this view. Finally, we can confirm that RelBERT outperforms the OpenAI API with GPT-3_{davinci} in all the datasets.

Google	Encyclopedic	Morphological	
$\text{RelBERT}_{\text{BASE}}$	93.0	86.3	
$\text{RelBERT}_{\text{LARGE}}$	98.6	92.6	
BATS	**Encyclopedic**	**Lexical**	**Morphological**
$\text{RelBERT}_{\text{BASE}}$	57.8	62.9	80.3
$\text{RelBERT}_{\text{LARGE}}$	71.3	72.4	90.0
SCAN	**Metaphor**	**Science**	
$\text{RelBERT}_{\text{BASE}}$	23.4	35.0	
$\text{RelBERT}_{\text{LARGE}}$	24.8	35.6	

Table 4.9: The accuracy of RelBERT on each domain of three analogy question datasets.

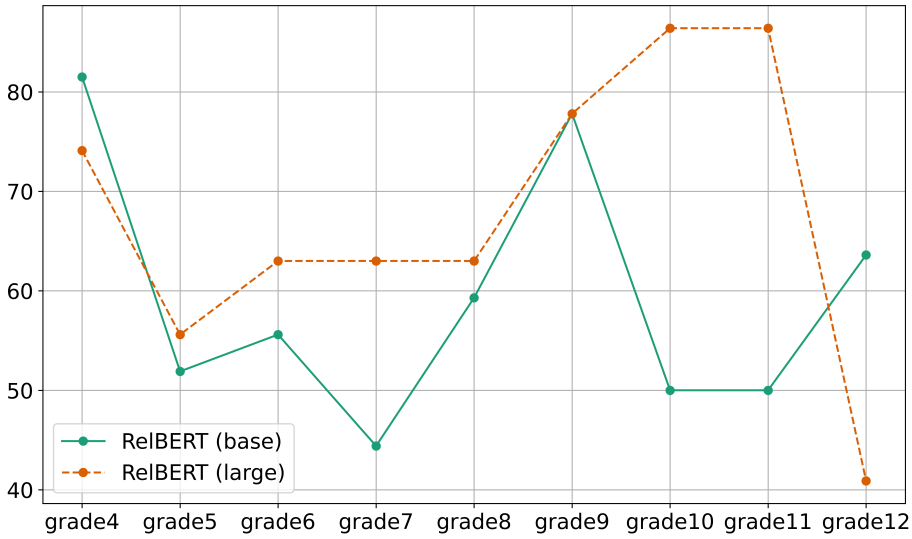

Figure 4.3: The accuracy of RelBERT for each domain of U2 analogy question.

4.5.2 Prediction Breakdown

We first group the accuracy of some analogy question datasets into their categories as shown in Table 4.4. Table 4.9 shows accuracy in each domain, and it clearly shows that both RelBERT models can achieve very high accuracy in the morphological relationship, despite not being trained on such relations. We can also con-

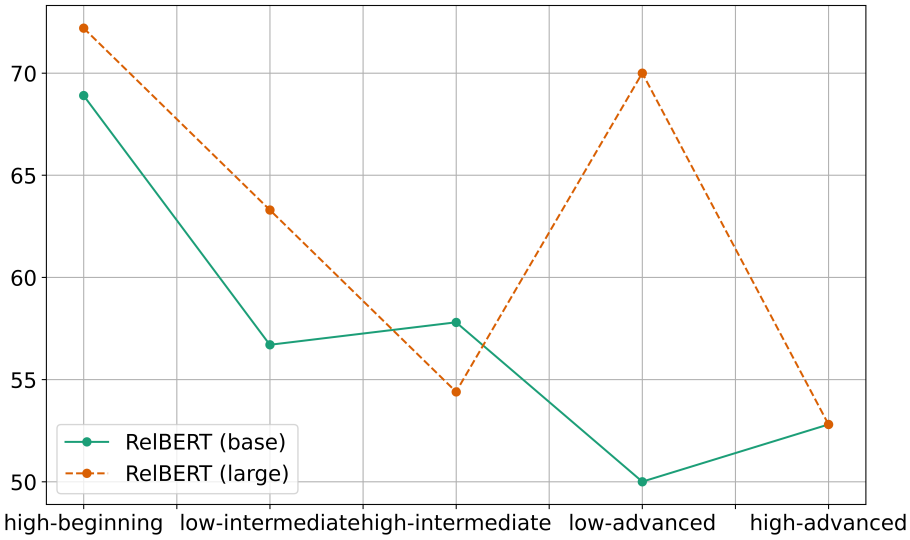

Figure 4.4: The accuracy of RelBERT for each domain of U4 analogy question.

firm such ability scales along with the model size, as $\text{RelBERT}_{\text{LARGE}}$ outperforms $\text{RelBERT}_{\text{BASE}}$ with around 10% in morphological relation in BATS, and 6% in morphological relation in Google. Figure 4.3 and Figure 4.4 show the accuracy along with the difficulty level in U2 and U4. Although we cannot see a clear signal in U2, we can see that models struggle more when the difficulty level is increased in U4, especially for $\text{RelBERT}_{\text{BASE}}$. Note that U2 is for the children, while U4 is for the college students, so it can be said that the ability of understanding relation in RelBERT is limited to certain levels, when it comes to an educational analogy question.

4.5.3 Lexical Relation Classification

Table 4.10 shows the micro F1 score for the lexical relation classification datasets. We can see that $\text{RelBERT}_{\text{LARGE}}$ is in general competitive with the SoTA approaches.

Model		BLESS	CogALexV	EVALution	K&H+N	ROOT09
GloVe	*cat*	93.3	73.5	58.3	94.9	86.5
	cat+dot	93.7	79.2	57.3	95.1	89.0
	diff	91.5	70.8	56.9	94.4	86.3
	diff+dot	92.9	78.5	57.9	94.8	88.9
fastText	*cat*	92.9	72.4	57.9	93.8	85.5
	cat+dot	93.2	77.4	57.8	94.0	88.5
	diff	91.2	70.2	55.5	93.3	86.0
	diff+dot	92.9	77.8	57.4	93.6	88.9
SoTA	LexNET	89.3	-	60.0	98.5	81.3
	SphereRE	**93.8**	-	62.0	**99.0**	86.1
$\text{RelBERT}_{\text{BASE}}$		90.0	83.7	64.2	94.0	88.2
$\text{RelBERT}_{\text{LARGE}}$		92.0	**85.0**	**68.4**	95.6	**90.4**

Table 4.10: Micro F1 score (%) for lexical relation classification.

For two (EVALution and ROOT09) out of the four lexical relation classification datasets that have SoTA results, $\text{RelBERT}_{\text{LARGE}}$ achieves the best results. Moreover, for these two datasets, even $\text{RelBERT}_{\text{BASE}}$ outperforms the SoTA methods. $\text{RelBERT}_{\text{LARGE}}$ outperforms the word embedding baselines in all datasets except for BLESS. We see a consistent improvement in accuracy when going from $\text{RelBERT}_{\text{BASE}}$ to $\text{RelBERT}_{\text{LARGE}}$.

4.6 Analysis

In this section, we analyze the capability of RelBERT from different aspects. We investigate the generalization ability of RelBERT for unseen relation in § 4.6.1. In § 4.6.2, we compare RelBERT with conversational LMs and few-shot learning. Then, we analyze the effect of the choice in the model architecture in § 4.6.3. Finally, in § 4.6.4 we present a qualitative analysis, where among others we show a visualization of the latent representation space of relation vectors. All the model

Dataset	Antonym	Attribute	Hypernym	Meronym	Synonym	*Full*
BLESS						
- Attribute	91.6	90.7	90.6	91.6	90.9	91.5
- Co-hyponym	94.6	95.3	95.5	94.0	93.8	93.5
- Event	84.1	84.2	84.0	82.2	84.1	83.6
- Hypernym	92.6	93.5	93.5	91.3	93.1	93.1
- Meronym	85.7	86.8	87.5	85.3	86.7	85.0
- Random	92.1	92.5	92.1	91.7	91.6	91.9
CogALexV						
- Antonym	60.5	64.0	62.9	67.9	63.3	68.2
- Hypernym	56.6	55.5	56.2	56.7	56.4	59.3
- Meronym	70.3	69.5	65.4	70.3	70.5	64.5
- Random	91.9	92.7	92.3	93.2	91.6	92.4
- Synonym	39.0	44.0	42.5	44.4	42.2	45.4
EVALution						
- Attribute	80.7	81.7	80.4	80.3	81.6	82.7
- Antonym	72.0	74.3	73.3	75.2	73.8	73.6
- Hypernym	57.7	59.3	58.5	60.3	59.1	57.5
- Meronym	68.3	71.6	69.0	64.5	66.9	68.8
- Possession	66.7	70.3	66.4	66.0	63.5	67.4
- Synonym	40.6	42.9	37.4	42.9	37.5	41.0
K&H+N						
- Co-hyponym	95.7	96.0	94.2	96.1	94.6	95.1
- Meronym	63.9	63.9	57.7	59.8	62.4	56.7
- Random	96.1	95.9	94.9	96.0	95.3	95.3
ROOT09						
- Co-hyponym	96.9	97.3	96.4	97.3	95.9	95.8
- Hypernym	80.3	80.3	79.0	81.8	79.2	79.5
- Random	89.7	89.7	89.3	89.8	89.0	88.8

Table 4.11: F1 score for each relation type of all the lexical relation classification datasets from $\text{RelBERT}_{\text{BASE}}$ models fine-tuned on the RelSim without a specific relation. The *Full* model on the right most column is the original $\text{RelBERT}_{\text{BASE}}$ model fine-tuned on full RelSim. The result of the relation type where the model is fine-tuned on RelSim without it, is emphasized by underline.

configurations of RelBERT can be found in § 4.6.5.

4.6.1 Analysis on Generalization Ability of RelBERT

RelBERT, when trained on RelSim, achieves competitive results on named entities (i.e. Nell-One and T-REX), despite the fact that RelSim does not contain any examples involving named entities. This is one of the most interesting aspect of RelBERT, where it leans to infer the relation based on the knowledge from the LM, instead of memorizing the word pairs in the training corpus. To understand the generalization ability of RelBERT more in depth, we conduct an additional experiment, where we explicitly exclude a specific relation from RelSim to train RelBERT and evaluate it on the excluded relation. We train a RelBERT on a number of variants of RelSim, where each time a specific relation type is excluded. We then test the resulting model on the lexical relation classification datasets. We focus this analysis on *Antonym*, *Attribute*, *Hypernym*, *Meronym*, and *Synonym*, which are shared between RelSim Table 4.1 and at least one of the lexical relation classification datasets Table 4.5. We train $\text{RelBERT}_{\text{BASE}}$ with InfoNCE on the different RelSim variants. Table 4.11 shows the results for six variants of RelBERT: give variants where a particular relation is excluded from RelSim and the model that was trained on the full RelSim training set. It can be observed that the the performance reduces by at most a few percentage points after removing a given target relation. In some cases, we can even see that the results improve after the removal. Hypernym is covered by all the datasets except K&H+N, and the largest decrease can be seen for CogALexV, which is around 3 percentage points. Meronym is covered by all the datasets except ROOT09. After removing the Meronym relation from the training data, the F1 score on meronym prediction increases in three out of four datasets. A similar pattern can be observed for the synonym relation, where the model that was trained without the synonym relation achieves better results

than the model trained on the full RelSim dataset. On the other hand, for antonym and attribute, we can see that removing these relations from the training data leads to somewhat lower results on these relations. The average F1 scores over all the relation types are also competitive with, and often even better than those for the full model. These results clearly support the idea that RelBERT can generalise beyond the relation types it is trained on.

4.6.2 Additional Baselines

Our main experiment focuses on zero-shot learning, where we do not assume any training or validation set. To compare the performance of RelBERT with more broader classes of models, we test conversational LMs via OpenAI in § 4.6.2.1, other type of prompt in § 4.6.2.3, and few-shot learning via a public large LM in § 4.6.2.2.

4.6.2.1 ChatGPT&GPT-4

ChatGPT (`gpt-3.5-turbo`)[14] and GPT-4 (`gpt-4`)[15] are two conversational LMs released by OpenAI as part of their product. Same as GPT-3, those models are private and we can only access through the API. Unlike GPT-3 however, we cannot obtain log-likelihood on each token to compute perplexity through the API, so we instead consider to ask those models directly with prompts and parse the output to get the prediction. We rely on the following two text prompts

1. Answer the question by choosing the correct option. Which of the

[14] https://openai.com/blog/chatgpt
[15] https://openai.com/research/gpt-4

following is an analogy?
1) A is to B what C_1 is to D_1
2) A is to B what C_2 is to D_2
3) A is to B what C_3 is to D_3
...
κ) A is to B what C_κ is to D_κ
The answer is

2. Only one of the following statements is correct. Please answer by choosing the correct option.
1) The relation between A and B is analogous to the relation between C_1 and D_1
2) The relation between A and B is analogous to the relation between C_2 and D_2
3) The relation between A and B is analogous to the relation between C_3 and D_3
...
κ) The relation between A and B is analogous to the relation between C_κ and D_κ
The answer is

where A and B is a query word pair, and $[C_i, D_i]_{i=1,...,\kappa}$ are the candidate word pairs. These prompts are used as inputs to the models to return a single reply, where we manually parse to get the prediction. As GPT-4 is the most expensive endpoint at the moment, we only report the accuracy on the SAT analogy question dataset.

Table 4.12 shows the result, and we can see that GPT-4 achieves SoTA with one

	ChatGPT	GPT-4
Prompt 1	34.7	62.5
Prompt 2	45.7	79.6

Table 4.12: Accuracy on SAT analogy question for ChatGPT and GPT-4 with the different prompts.

Random Seed	0	1	2	3	4	Average
1-shot	44.4	44.7	40.1	48.4	46.0	44.7
5-shots	46.0	47.1	39.8	44.9	49.5	45.5
10-shots	45.7	48.9	33.7	39.6	45.5	42.7

Table 4.13: The accuracy of [1, 5, 10]-shots learning with five different random seeds.

of the prompt and ChatGPT is worse than GPT-4. However, the gap between two prompts are more than 15%, which shows its sensitivity to the prompt. Note that as our main experiment focuses on zero-shot learning, these results are not compatible with the zero-shot baselines in § 4.5.1, as we assume to have no validation set, where one could choose the best prompt. Note that as our main experiment focuses on zero-shot learning, these results are not compatible with RelBERT, as we assume to have no validation set, where one could choose the best prompt.

4.6.2.2 Few-shot Learning

In the main experiment (§ 4.5), we used the LM baselines in a zero-shot setting. However, recent large LMs often perform better when a few examples are provided as part of the input (Brown et al., 2020; Chung et al., 2022; Iyer et al., 2022). The idea is to provide a few (input,output) pairs at the start of the prompt, followed by the target input. This strategy is commonly referred to as few-shot learning or in-context learning. It is most effective for larger LMs, which can recognize the pat-

tern in the (input,output) pairs and apply this pattern to the target input (Brown et al., 2020; Chung et al., 2022; Iyer et al., 2022). Since RelBERT is fine-tuned on RelSim, for this experiment we provide example pairs to the LM input which are taken from RelSim as well.

We focus on the SAT benchmark and the Flan-T5$_{\text{XXL}}$ model, which was the best-performing LM on SAT in the main experiments. We consider [1, 5, 10]-shot learning. The demonstrations in each experiment are randomly chosen from the training split of RelSim. We use the same template as for the zero-shot learning, both to describe the examples and to specify the target input. For example, in 5-shot learning setting, a complete input to the model with five demonstrations of $[\hat{A}_i, \hat{B}_i, \hat{C}_i, \hat{D}_i]_{i=1...5}$ and the target query of $[A, B]$ is shown as below.

> $\hat{A}_1$ is to $\hat{B}_1$ what $\hat{C}_1$ is to $\hat{D}_1$
> $\hat{A}_2$ is to $\hat{B}_2$ what $\hat{C}_2$ is to $\hat{D}_2$
> $\hat{A}_3$ is to $\hat{B}_3$ what $\hat{C}_3$ is to $\hat{D}_3$
> $\hat{A}_4$ is to $\hat{B}_4$ what $\hat{C}_4$ is to $\hat{D}_4$
> $\hat{A}_5$ is to $\hat{B}_5$ what $\hat{C}_5$ is to $\hat{D}_5$
> A is to B what

We run each experiment for five different random seeds (i.e. five different few-shot prompts for each setting). Table 4.13 shows the result. Somewhat surprisingly, the few-shot models consistently perform worse than the zero-shot model, which achieved an accuracy of 52.4 in the main experiment.

	Flan-T5$_{\text{XXL}}$	Flan-UL2	OPT-IML$_{\text{30B}}$	OPT-IML$_{\text{M-30B}}$
Analogical Statement	52.4	50.0	48.9	48.9
Multi-choice QA	35.8	40.6	27.3	31.3

Table 4.14: The accuracy with multiple-choice prompting compared to the vanilla prompting strategy with the analogical statement ("A is to B what C is to D") on SAT.

4.6.2.3 Multi-choice Prompt

In our main experiment, we compute perplexity separately on each candidate for LMs, but the analogy question can be formatted as a multiple-choice question answering prompt as well. Such prompt has a benefit by receiving all the candidates at once, but it requires LMs to understand the question properly. Following a typical template to solve multiple-choice QA in the zero-shot setting (Brown et al., 2020; Chung et al., 2022; Iyer et al., 2022), we create following text prompt

> Which of the following is an analogy?
> 1) A is to B what C_1 is to D_1
> 2) A is to B what C_2 is to D_2
> 3) A is to B what C_3 is to D_3
> ...
> κ) A is to B what C_κ is to D_κ
> The answer is

where A and B is a query word pair, and $[C_i, D_i]_{i=1,\ldots,\kappa}$ are the candidate word pairs. Table 4.14 shows the accuracy on SAT for the top-4 LMs in SAT with the original analogical statement prompt, and we can confirm that the multiple-choice prompt is worse in all the LMs with large drop.

	Dataset	RelSim	NELL	T-REX	ConceptNet
Analogy Question	SAT	**59.9**	36.1	46.8	44.9
	U2	**59.6**	39.9	42.5	42.5
	U4	**57.4**	41.0	44.0	41.0
	BATS	**70.3**	45.5	51.0	62.0
	Google	**89.2**	67.8	75.0	81.0
	SCAN	**25.9**	14.9	19.6	21.8
	NELL	62.0	**82.5**	72.2	66.2
	T-REX	66.7	69.9	**83.6**	44.8
	ConceptNet	**39.8**	10.3	18.8	22.7
	Average	**59.0**	45.3	50.4	47.4
Classification	BLESS	**90.0**	88.6	89.3	88.8
	CogALexV	**83.7**	78.6	82.8	82.8
	EVALution	64.2	58.7	**64.7**	62.8
	K&H+N	94.0	94.8	**95.2**	95.0
	ROOT09	88.2	86.6	88.2	**88.8**
	Average	84.0	81.5	**84.1**	83.6

Table 4.15: The results on analogy questions (accuracy) and lexical relation classification (micro F1 score) of RelBERT with different training datasets, where the best result across models in each dataset are shown in bold.

4.6.3 Ablation Analysis

In this section, we analyze how the performance of RelBERT depends on different design choices that were made. We look at the impact of the training dataset in § 4.6.3.1; the loss function in § 4.6.3.2; the number of negative samples for the InfoNCE loss in § 4.6.3.3; the base LM in § 4.6.3.4; the prompt templates in § 4.6.3.5; and the impact of random variations in § 4.6.3.6. Throughout this section, we use $\text{RoBERTa}_{\text{BASE}}$ for efficiency.

4.6.3.1 The Choice of Datasets

RelSim is relatively small and does not cover named entities, though the RelBERT trained on RelSim, performs the best in T-REX and NELL-One, the named-entity

analogy questions in our main result. Here we present a comparison with a number of alternative training sets, to see whether better results might be possible. We are primarily interested to see whether the performance on NELL and T-REX might be improved by training RelBERT on the training splits of these datasets. We fine-tune RoBERTa$_{\text{BASE}}$ on three datasets introduced in § 4.2.3: NELL-One, T-REX and ConceptNet. We use InfoNCE in each case. The results are summarised in Table 4.15. We can see that training RelBERT on RelSim leads to the best results on most datasets, and the best result on average by a large margin. This is despite the fact that RelSim is significantly smaller than the other datasets (see Table 4.2). It is particularly noteworthy that training on RelSim outperforms training on ConceptNet even on the ConceptNet test set, even though ConceptNet contains several relation types that are not covered by RelSim. However, when it comes to the relationships between named entities, and the NELL and T-REX benchmarks in particular, training on RelSim underperforms training on NELL or T-REX. Overall, we find that training RelBERT on RelSim is the key to obtain the generalization ability, especially as seen in the superior accuracy in ConceptNet to the RelBERT trained on ConceptNet, but it cannot fully understand named-entity solely with RelSim.

4.6.3.2 The Choice of Loss Function

In this section, we compare the performance of three different loss functions for training RelBERT. In particular, we fine-tune RoBERTa$_{\text{BASE}}$ on RelSim, and we consider the triplet loss and InfoLOOB, in addition to InfoNCE (see § 4.4.1 for more in detail). Table 4.16 shows the result of RelBERT fine-tuned with each of the loss functions. We can see that none of the loss functions consistently outper-

	Dataset	Triplet	InfoNCE	InfoLOOB
Analogy Question	SAT	54.5	**59.9**	58.8
	U2	55.3	**59.6**	57.5
	U4	**58.6**	57.4	56.3
	BATS	**72.6**	70.3	67.6
	Google	86.4	**89.2**	83.8
	SCAN	**29.5**	25.9	27.0
	NELL	**70.7**	62.0	67.5
	T-REX	45.4	**66.7**	65.6
	ConceptNet	29.4	39.8	**40.0**
	Average	55.8	**59.0**	58.2
Classification	BLESS	88.7	90.0	**91.0**
	CogALexV	80.5	**83.7**	83.3
	EVALution	**67.7**	64.2	65.8
	K&H+N	93.1	94.0	**94.9**
	ROOT09	**90.3**	88.2	89.3
	Average	84.1	84.0	**84.9**

Table 4.16: The results on analogy questions (accuracy) and lexical relation classification (micro F1 score) with different loss functions, where the best result across models in each dataset is shown in bold.

forms the other. On average, InfoNCE achieves the best results on the analogy questions. The difference with InfoLOOB is small, which is to be expected given that InfoNCE and InfoLOOB are closely related. While the triple loss performs worse on average, it still manages to achieve the best results in four out of nine analogy datasets. For the relation classification experiments, the results are much closer, with InfoNCE now performing slightly worse than the other loss functions.

4.6.3.3 The Choice of the Number of Negative Samples

The variant of InfoNCE that we considered for training RelBERT relies on in-batch negative samples, i.e. the negative samples for a given anchor pair correspond to the other word pairs that are included in the same batch. The number of negative samples that are considered thus depends on the batch size. In general, using

	Batch Size	25	50	100	150	200	250	300	350	400	450	500
Analogy Question	SAT	56.1	59.1	53.2	55.6	56.4	57.2	57.5	58.3	**59.9**	57.2	57.2
	U2	55.3	56.1	46.1	53.9	56.1	**60.1**	56.1	56.6	59.6	59.6	55.7
	U4	56.5	57.4	52.5	54.9	57.2	**59.0**	57.2	54.2	57.4	58.3	56.5
	BATS	71.7	69.6	66.9	72.3	69.7	70.9	72.0	**73.7**	70.3	69.2	70.8
	Google	88.2	87.4	78.6	88.0	**90.2**	89.6	86.4	86.2	89.2	86.6	89.8
	SCAN	25.0	27.2	26.2	30.9	26.5	25.3	28.8	**31.6**	25.9	25.7	25.1
	NELL	67.0	66.5	71.5	**77.7**	66.0	63.7	75.0	**77.7**	62.0	62.8	63.5
	T-REX	59.6	60.7	57.9	57.9	62.3	60.1	57.9	60.1	66.7	**69.4**	62.8
	ConceptNet	36.9	39.3	31.1	29.4	40.5	39.6	31.0	31.4	39.8	38.9	**42.8**
	Average	57.4	58.1	53.8	57.8	58.3	58.4	58.0	58.9	**59.0**	58.6	58.2
Classification	BLESS	90.3	90.7	90.3	90.6	90.6	89.5	**91.3**	90.6	89.6	90.4	89.7
	CogALexV	66.0	**67.9**	65.9	62.5	65.2	64.7	65.4	64.4	65.8	65.4	63.6
	EVALution	64.8	62.1	**65.0**	62.9	63.6	64.6	63.8	63.2	62.9	62.8	64.6
	K&H+N	86.0	84.8	87.0	**87.5**	85.3	85.2	85.2	87.2	84.6	85.4	85.1
	ROOT09	87.7	88.8	87.7	88.6	**89.4**	88.8	88.8	88.6	87.9	88.2	88.0
	Average	79.0	78.9	**79.2**	78.4	78.8	78.6	78.9	78.8	78.2	78.4	78.2

Table 4.17: The results on analogy questions (accuracy) and lexical relation classification (micro F1 score) with different batch size (negative samples) at InfoNCE, where the best result across models in each dataset is shown in bold.

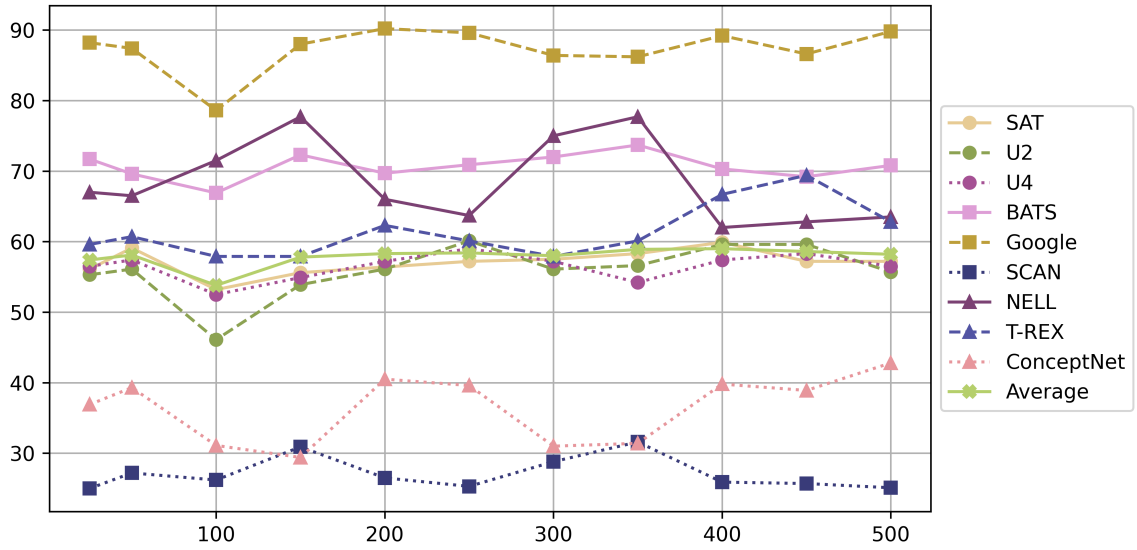

Figure 4.5: The results of analogy questions along with the batch size.

a larger number of negative samples tends to benefit contrastive learning strategies, but it comes at the price of an increase in memory requirement. Here we analyse

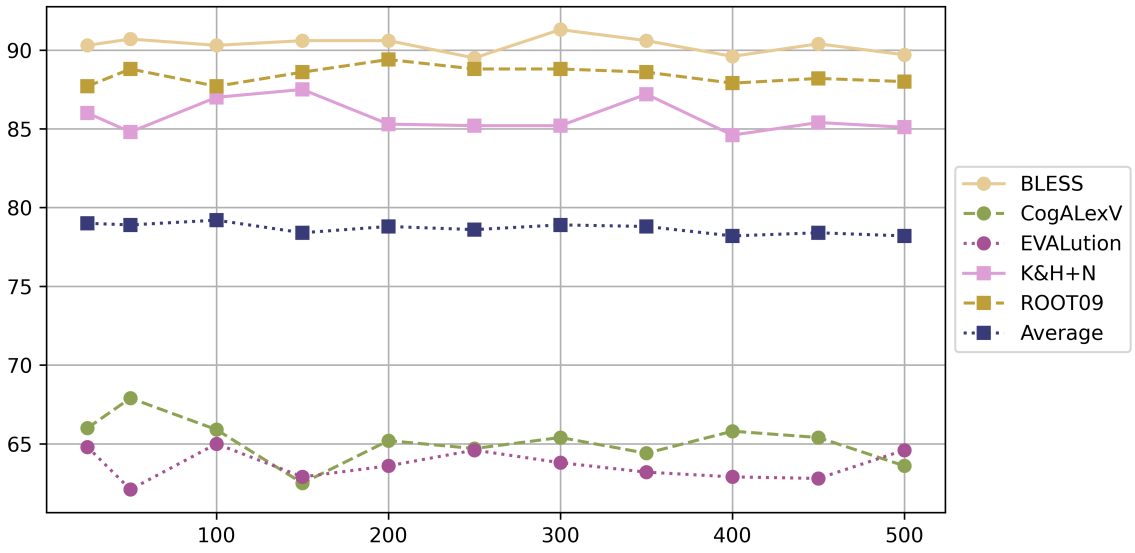

Figure 4.6: The results of lexical relation classification along with the batch size.

the impact of this choice, by comparing the results we obtained for different batch sizes. We train $\text{RelBERT}_{\text{BASE}}$ on RelSim with batch sizes from [25, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500], where the batch size 400 corresponds to our main $\text{RelBERT}_{\text{BASE}}$ model. The results are shown in Table 4.17, and visually illustrated in Figure 4.5 and Figure 4.6. Somewhat surprisingly, the correlation between batch size and performance is very weak. For analogy questions, there is a weak positive correlation. The Spearman's correlation to the batch size for T-REX is 0.6 with p-value 0.047, but in other datasets, correlations are not significant (p-values are higher than 0.05). Indeed, even a batch size of 25 is sufficient to achieve close-to-optimal results. For lexical relation classification, the correlation is even weaker, and none of the dataset shows significant Spearman's correlation coefficient.

		$\text{BERT}_{\text{BASE}}$	$\text{ALBERT}_{\text{BASE}}$	$\text{RoBERTa}_{\text{BASE}}$
Analogy Question	SAT	44.7	40.4	**59.9**
	U2	36.8	35.5	**59.6**
	U4	40.0	38.7	**57.4**
	BATS	54.9	59.2	**70.3**
	Google	72.2	56.4	**89.2**
	SCAN	23.7	21.2	**25.9**
	NELL	56.7	47.7	**62.0**
	T-REX	49.2	32.8	**66.7**
	ConceptNet	27.1	25.7	**39.8**
	Average	45.0	39.7	**59.0**
Classification	BLESS	**90.9**	88.0	90.0
	CogALexV	80.7	78.3	**83.7**
	EVALution	61.8	58.1	**64.2**
	K&H+N	**95.5**	92.9	94.0
	ROOT09	**88.7**	85.6	88.2
	Average	83.5	80.6	**84.0**

Table 4.18: The results on analogy questions (accuracy) and lexical relation classification (micro F1 score) of RelBERT with different LMs, where the best results across models in each dataset are shown in bold.

4.6.3.4 The Choice of Language Model

Thus far, we have only considered RoBERTa as the base language model for training RelBERT. Here we compare RoBERTa with two alternative choices: BERT and ALBERT (Lan et al., 2019). We compare $\text{BERT}_{\text{BASE}}$[16], $\text{ALBERT}_{\text{BASE}}$[17], and $\text{RoBERTa}_{\text{BASE}}$[18], fine-tuned on RelSim with InfoNCE. Table 4.18 shows the result. $\text{RoBERTa}_{\text{BASE}}$ is found to consistently achieve the best results on analogy questions, with a surprisingly large margin. $\text{RoBERTa}_{\text{BASE}}$ also achieved the best result, on average, for lexical relation classification, although in this case it only achieves the best results in two out of five datasets. ALBERT consistently has the worst performance, struggling even on the relatively easy Google dataset. These results

[16] https://huggingface.co/bert-base-cased
[17] https://huggingface.co/albert-base-v2
[18] https://huggingface.co/roberta-base

clearly show that the choice of the LM is of critical importance for the performance of RelBERT.

4.6.3.5 The Choice of Prompt Template

Our main experiment relies on the five prompt templates introduced in § 4.2.1, where we choose the best among these five templates based on the validation loss. We now analyse the impact of these prompt templates. We focus this analysis on $\text{RelBERT}_{\text{BASE}}$, i.e. $\text{RoBERTa}_{\text{BASE}}$ fine-tuned with InfoNCE on RelSim. For this configuration, the template that was selected based on validation accuracy is

> Today, I finally discovered the relation between [**h**] and [**t**] : [**h**] is the `<mask>` of [**t**]

We experiment with a number of variations of this template. First, we will see whether the length of the template plays an important role, and in particular whether a similar performance can be achieved with shorter templates. Subsequently we also analyse to what extent the wording of the template matters, i.e. whether similar results are possible with templates that are less semantically informative.

The Effect of Length. We start from the best template chosen for $\text{RelBERT}_{\text{BASE}}$, and shorten it while preserving its meaning as much as possible. Specifically, we considered the following variants:

1. Today, I finally discovered the relation between [**h**] and [**t**] : [**h**] is the `<mask>` of [**t**]

	Template	1 (Original)	2	3	4	5
Analogy Question	SAT	**59.9**	58.3	56.7	59.4	46.3
	U2	**59.6**	57.9	57.9	56.6	44.7
	U4	57.4	57.6	54.9	**60.0**	46.8
	BATS	70.3	69.6	70.0	**73.9**	65.6
	Google	89.2	88.0	89.4	**93.4**	81.8
	SCAN	25.9	24.8	23.9	**27.1**	25.2
	NELL	62.0	**65.5**	64.2	**65.5**	59.2
	T-REX	**66.7**	62.3	60.1	56.8	48.1
	ConceptNet	**39.8**	39.0	37.8	39.5	32.4
	Average	59.0	59.4	58.8	**61.7**	51.7
Classification	BLESS	89.9	89.2	89.2	**90.5**	88.2
	CogALexV	65.7	65.3	66.7	**69.6**	63.3
	EVALution	**65.1**	64.8	63.1	64.9	63.0
	K&H+N	85.3	85.3	83.7	86.2	**86.9**
	ROOT09	89.1	87.6	88.9	**89.8**	87.8
	Average	79.0	78.4	78.3	**80.2**	77.8

Table 4.19: The results on analogy questions (accuracy) and lexical relation classification (micro F1 score) of RelBERT fine-tuned with different length of templates, where the best results across models in each dataset are shown in bold.

2. I discovered the relation between [**h**] and [**t**]: [**h**] is the `<mask>` of [**t**]
3. the relation between [**h**] and [**t**]: [**h**] is the `<mask>` of [**t**]
4. I discovered: [**h**] is the `<mask>` of [**t**]
5. [**h**] is the `<mask>` of [**t**]

For each of the templates, we fine-tune RoBERTa$_{\text{BASE}}$ with InfoNCE on RelSim. The results are summarised in Table 4.19. We find that template 4 outperforms the original template 1 on average, both for analogy questions and for lexical relation classification. In general, we thus find no clear link between the length of the template and the resulting performance, although the shortest template (template 5) achieves by far the worst results. This suggests that, while longer templates are not necessarily better, using templates which are too short may be problematic.

	Phrase	*the spaceship*	*Napoleon Bonaparte*	*football*	*Italy*	*Cardiff*	*the earth science*
Analogy Question	SAT	57.2	56.7	56.1	58.0	57.2	59.6
	U2	55.3	58.3	56.6	55.3	57.5	58.3
	U4	56.9	**58.1**	56.5	56.2	55.1	56.9
	BATS	**71.2**	69.1	69.5	68.8	69.5	69.9
	Google	87.2	85.4	87.6	85.2	86.8	**89.2**
	SCAN	25.6	**26.8**	25.7	26.1	26.2	22.6
	NELL	64.8	61.7	63.8	63.5	60.0	64.7
	T-REX	63.4	51.9	57.4	59.6	55.7	56.3
	ConceptNet	39.6	39.4	38.5	36.9	37.9	39.3
	Average	57.9	56.4	56.9	56.6	56.2	57.4
Classification	BLESS	89.9	89.7	90.0	90.0	89.2	**91.4**
	CogALexV	63.4	64.7	**66.5**	65.7	66.3	65.3
	EVALution	63.6	63.7	64.0	63.8	62.4	63.5
	K&H+N	84.8	86.2	**86.4**	85.9	85.3	84.8
	ROOT09	88.5	88.9	89.4	89.0	89.2	88.7
	Average	78.0	78.6	79.3	78.9	78.5	78.7

	Phrase	*pizza*	*subway*	*ocean*	*Abraham Lincoln*	*the relation* (Original)
Analogy Question	SAT	57.0	59.6	56.7	**59.9**	**59.9**
	U2	55.7	57.0	57.5	56.1	**59.6**
	U4	56.7	55.3	55.8	57.2	57.4
	BATS	68.5	69.8	68.9	69.5	70.3
	Google	85.6	87.6	86.0	**89.2**	**89.2**
	SCAN	25.8	26.6	25.6	24.6	25.9
	NELL	65.8	63.3	63.3	**66.2**	62.0
	T-REX	59.0	60.1	60.7	60.7	**66.7**
	ConceptNet	38.5	38.8	38.8	38.3	**39.8**
	Average	57.0	57.6	57.0	58.0	**59.0**
Classification	BLESS	89.7	89.7	90.6	89.3	89.9
	CogALexV	66.0	65.3	65.3	64.1	65.7
	EVALution	64.5	63.3	63.5	64.0	**65.1**
	K&H+N	84.7	84.6	85.1	85.0	85.3
	ROOT09	89.1	89.5	88.9	**89.6**	89.1
	Average	78.8	78.5	78.7	78.4	**79.0**

Table 4.20: The results on analogy questions (accuracy) and lexical relation classification (micro F1 score) of RelBERT fine-tuned with random phrase to construct the template, where the best results across models in each dataset are shown in bold.

The Effect of Semantics. We now consider variants of the original template in which the anchor phrase "the relation" is replaced by a semantically meaningful

distractor, i.e. we consider templates of the following form:

> Today, I finally discovered `<semantic phrase>` between **[h]** and **[t]** : **[h]** is the `<mask>` of **[t]**

where `<semantic phrase>` is a placeholder for the chosen anchor phrase. We randomly chose 10 phrases (four named entities and six nouns) to play the role of this anchor phrase. For each of the resulting templates, we fine-tune $\text{RoBERTa}_{\text{BASE}}$ with InfoNCE on RelSim dataset. Table 4.20 shows the result. We can see that the best results are obtained with the original template, both for analogy questions and for lexical relation classification. Nevertheless, the difference in performance is surprisingly limited. The largest decrease is 2.8 in the average for analogy questions and 1.0 in the average for lexical relation classification, which is smaller than the differences we observed when using the shortest template, or when changing the LM § 4.6.3.4 or the loss function § 4.6.3.2.

4.6.3.6 The Choice of Random Seed

In this section, we investigate the stability of RelBERT training, by comparing the results we obtained for different random seeds. We use a fixed random seed of 0 as default in the main experiments. Here we include results for two other choices of the random seed. We train both of $\text{RelBERT}_{\text{BASE}}$ and $\text{RelBERT}_{\text{LARGE}}$. However, different as for the main results, for this analysis we reduce the batch size from 400 to 100 for $\text{RelBERT}_{\text{LARGE}}$, to reduce the computation time. Table 4.21 shows the result. One thing that can be observed is that the standard deviation is higher for $\text{RelBERT}_{\text{BASE}}$ than for $\text{RelBERT}_{\text{LARGE}}$. For example, the accuracy of T-REX dif-

		RelBERT$_{\text{BASE}}$				RelBERT$_{\text{LARGE}}$			
	Random Seed	0	1	2	Average	0	1	2	Average
Analogy Question	SAT	59.9	54.3	55.9	56.7 ±2.9	68.2	68.4	71.9	69.5 ±2.1
	U2	59.6	51.8	55.7	55.7 ±3.9	67.5	65.4	68.0	67.0 ±1.4
	U4	57.4	53.7	54.4	55.2 ±2.0	63.9	65.0	66.4	65.1 ±1.3
	BATS	70.3	65.3	67.3	67.6 ±2.5	78.3	79.7	80.4	79.5 ±1.1
	Google	89.2	79.4	85.6	84.7 ±5.0	93.4	93.4	94.8	93.9 ±0.8
	SCAN	25.9	25.9	23.3	25.1 ±1.5	25.9	27.0	29.1	27.4 ±1.6
	NELL	62.0	71.0	63.5	65.5 ±4.8	60.5	67.3	66.3	64.7 ±3.7
	T-REX	66.7	55.2	46.4	56.1 ±10.1	67.8	65.0	63.4	65.4 ±2.2
	ConceptNet	39.8	27.4	30.5	32.6 ±6.4	43.3	44.8	48.7	45.6 ±2.8
	Average	59.0	53.8	53.6	55.5 ±3.1	63.2	64.0	65.4	64.2 ±1.1
Classification	BLESS	90.0	91.4	90.7	90.7 ±0.7	91.5	92.4	91.7	91.9 ±0.5
	CogALexV	83.7	81.5	81.1	82.1 ±1.4	84.9	86.5	86.4	85.9 ±0.9
	EVALution	64.2	63.3	63.1	63.5 ±0.6	66.9	69.0	67.8	67.9 ±1.0
	K&H+N	94.0	94.7	94.3	94.3 ±0.4	95.1	95.3	95.7	95.3 ±0.3
	ROOT09	88.2	88.3	89.5	88.7 ±0.7	89.2	89.5	91.5	90.1 ±1.3
	Average	84.0	83.8	83.7	83.9 ±0.1	85.5	86.5	86.6	86.2 ±0.6

Table 4.21: The result of RelBERT$_{\text{BASE}}$ and RelBERT$_{\text{LARGE}}$ with three runs with different random seed, and the average and the standard deviation in each dataset.

fers from 46.4 to 66.7 for RelBERT$_{\text{BASE}}$, while only ranging between 63.4 and 67.8 RelBERT$_{\text{LARGE}}$. We can also see that there is considerably less variation in performance for lexical relation classification, compared to analogy questions.

4.6.4 Qualitative Analysis

We present a qualitative analysis of the latent space of the RelBERT relation vectors. For this analysis, we focus on the test splits of the ConceptNet and NELL-One datasets. We compute the relation embeddings of all the word pairs in these datasets using RelBERT$_{\text{BASE}}$ and RelBERT$_{\text{LARGE}}$. As a comparison, we also compute relation embeddings using fastText, in the same way as in § 3.3.3. First, we visualize the relation embeddings, using t-Distributed Stochastic Neighbor Embedding (tSNE) (van der Maaten and Hinton, 2008) to map the embeddings to a two-

		AtLocation	CapableOf
RelBERT$_{\text{LARGE}}$	1	child:school, animal:zoo, prisoner:jail, fish:aquarium, librarian:library	dog:bark, cat:hunt mouse lawyer:settle lawsuit, pilot:land plane
	2	computer:office, book:shelf, coat:closet, food:refrigerator, paper clip:desk	knife:spread butter, clock:tell time, comb:part hair, match:light fire
RelBERT$_{\text{BASE}}$	1	animal:zoo, elephant:zoo, student:classroom, student:school	dog:bark, student:study, ball:bounce, tree:grow, bomb:explode
	2	computer:office, coat:closet, mirror:bedroom, notebook:desk, food:refrigerator	plane:fly, clock:tell time, computer:compute, knife:spread butter
fastText	1	fish:water, child:school, bookshelf:library, feather:bird, computer:office	cat:hunt mouse, cat:drink water, dog:guide blind person, dog:guard house
	2	animal:zoo, elephant:zoo, lion:zoo, weasel:zoo, tiger:zoo	knife:spread butter, turtle:live long time, magician:fool audience
		IsA	
RelBERT$_{\text{LARGE}}$	1	baseball:sport, sushi:food, yo-yo:toy, dog:mammal, spanish:language	
	2	canada:country, california:state, san francisco:city	
RelBERT$_{\text{BASE}}$	1	baseball:sport, soccer:sport, chess:game	
	2	dog:mammal, rose:flower, violin:string instrument, fly:insect, rice:food	

Table 4.22: Examples of word pairs in the clusters obtained by HDBSCAN for different relation embeddings. For the *IsA* relation, all word pairs were clustered together in the case of fastText.

dimensional space. Figure 4.7 and Figure 4.8 show the resulting two-dimensional relation embeddings for ConceptNet and Nell-One respectively. The plots clearly show how the different relation types are separated much more clearly for RelBERT

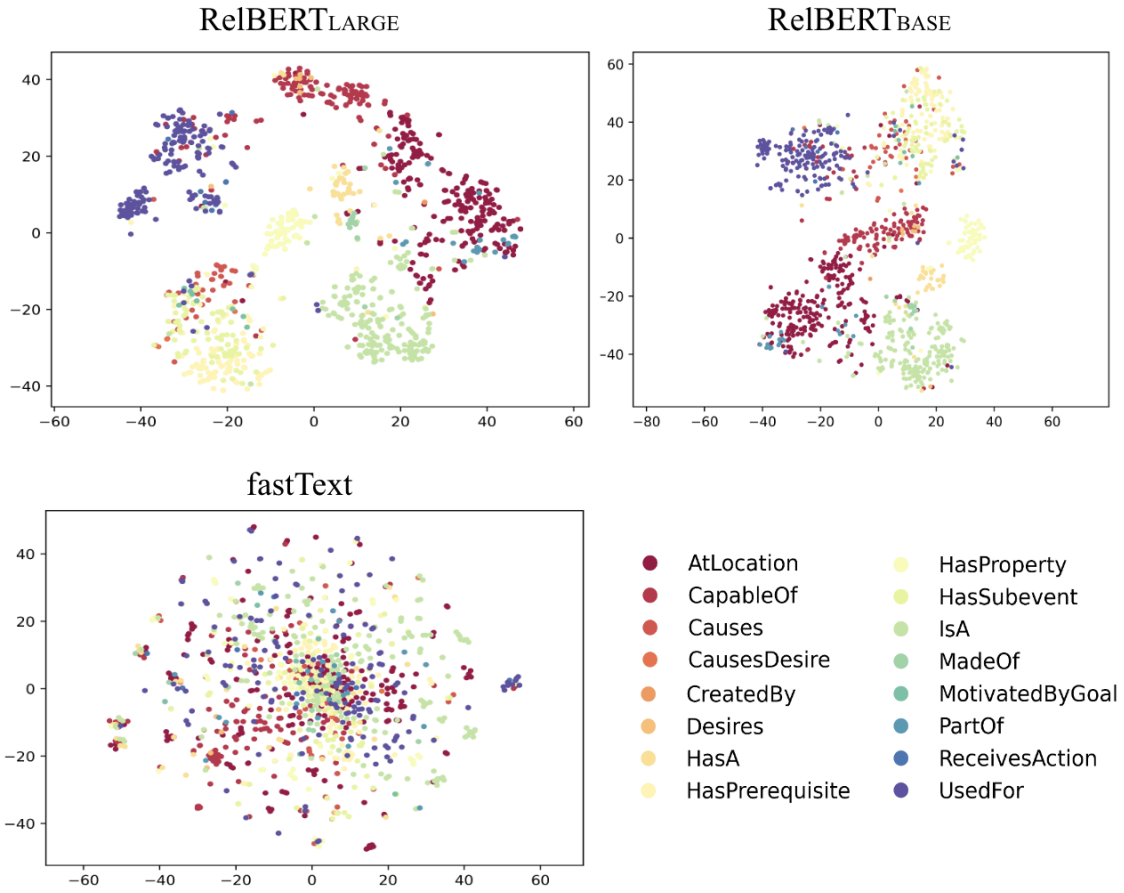

Figure 4.7: The tSNE 2-dimension visualization of relation embeddings over the test set of ConceptNet. Colours reflect the relation type of the embedded word pairs.

than for fastText. For ConceptNet, in particular, we can see that the fastText representations are mixed together. Comparing $\text{RelBERT}_{\text{LARGE}}$ and $\text{RelBERT}_{\text{BASE}}$, there is no clear difference for NELL-One. For ConceptNet, we can see that $\text{RelBERT}_{\text{LARGE}}$ leads to clusters which are somewhat better separated.

Second, we want to analyse whether RelBERT vectors could model relations in a more fine-grained way than existing knowledge graphs. We focus on ConceptNet

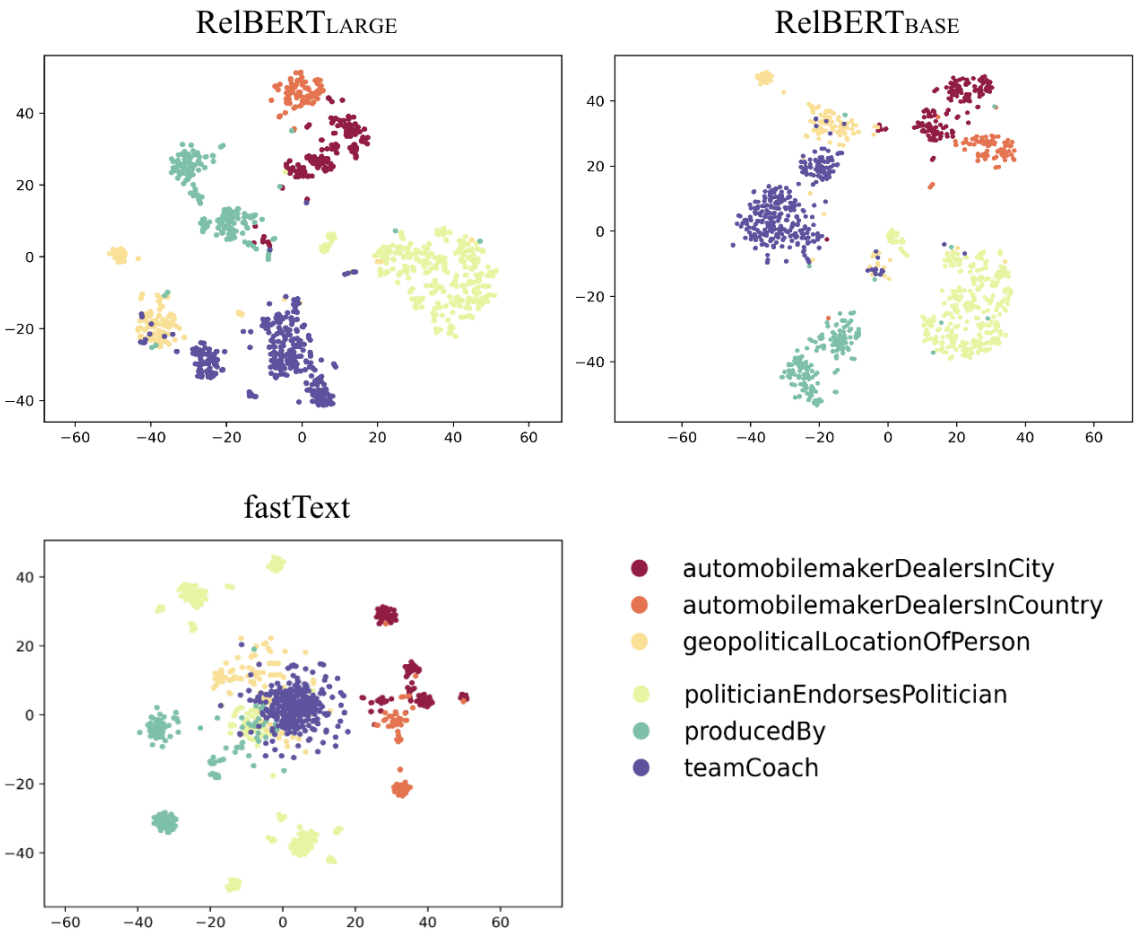

Figure 4.8: The tSNE 2-dimension visualization of relation embeddings over the test set of NELL-One. Colours reflect the relation type of the embedded word pairs.

for this analysis. We cluster the RelBERT vectors of the word pairs in each relation type using hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) (Campello et al., 2013). We focus on three relation types: *AtLocation*, *CapableOf*, and *IsA*. These are the relation types with the highest number of instances, among those for which HDBSCAN yielded more than one cluster. We obtained two clusters for each of these relation types. Table 4.22 shows some examples of word pairs in each cluster. For *AtLocation*, RelBERT separates the word

pairs depending on whether the head denotes a living thing. On the other hand, fastText captures a surface feature and forms a cluster where the word pairs have "zoo" as tails. All other word pairs are mixed together in the first cluster. For *CapableOf*, RelBERT$_{\text{LARGE}}$ again distinguishes the word pairs based on whether the head entity denotes a living thing. For RelBERT$_{\text{BASE}}$, the clusters are not separated as clearly, while fastText again focuses on the presence of particular words such as "cat" and "dog" in this case. For *IsA*, RelBERT$_{\text{LARGE}}$ yields a cluster that specifically focuses on geolocations. RelBERT$_{\text{BASE}}$ puts pairs with the words "sport" or "game" together. In the case of fastText, all pairs were clustered together. Overall, fastText tends to catch the surface features of the word pairs. RelBERT$_{\text{LARGE}}$ seems to find meaningful distinctions, although at least in these examples, they are focused on the semantic types of the entities involved rather than any specialisation of the relationship itself. The behaviour of RelBERT$_{\text{BASE}}$ is similar, albeit clearly noisier.

4.6.5 Hyperparameters

Table 4.23 shows the configuration we used in each experiment through the analysis.

4.7 Summary

We have proposed a strategy for learning relation embeddings, i.e. vector representations of pairs of words which capture their relationship. The main idea is to fine-tune a pre-trained LM using the relational similarity dataset from SemEval

Language Model	Dataset	Loss	Template	Epoch	Seed	Batch
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	1	8	0	400
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	5	10	1	400
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	5	9	2	400
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	1	10	0	25
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	1	6	0	50
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	2	6	0	100
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	5	8	0	150
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	1	8	0	200
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	1	9	0	250
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	5	10	0	300
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	5	9	0	350
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	1	8	0	450
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	1	9	0	500
$\text{RoBERTa}_{\text{LARGE}}$	RelSim	InfoNCE	1	8	0	100
$\text{RoBERTa}_{\text{LARGE}}$	RelSim	InfoNCE	4	9	1	100
$\text{RoBERTa}_{\text{LARGE}}$	RelSim	InfoNCE	4	8	2	100
$\text{RoBERTa}_{\text{LARGE}}$	RelSim	InfoNCE	4	9	0	400
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoLOOB	1	4	0	400
$\text{RoBERTa}_{\text{BASE}}$	RelSim	Triplet	5	1	0	400
$\text{RoBERTa}_{\text{BASE}}$	NELL	InfoNCE	5	5	0	400
$\text{RoBERTa}_{\text{BASE}}$	T-REX	InfoNCE	1	4	0	400
$\text{RoBERTa}_{\text{BASE}}$	ConceptNet	InfoNCE	4	5	0	400
$\text{RoBERTa}_{\text{BASE}}$	RelSim	InfoNCE	1	8	0	400
$\text{BERT}_{\text{BASE}}$	RelSim	InfoNCE	1	6	0	400
$\text{ALBERT}_{\text{BASE}}$	RelSim	InfoNCE	1	5	0	400

Table 4.23: The best configuration of the template and the number of epoch used in the analysis.

2012 Task 2, which covers a broad range of semantic relations. In our experimental results, we found the resulting relation embeddings to be of high quality, outperforming state-of-the-art methods on all the analogy questions and some of the relation classification benchmarks. Crucially, we found that RelBERT is capable of modelling relationships that go well beyond those that are covered by the training data, including morphological relations and relations between named entities. Being based on $\text{RoBERTa}_{\text{LARGE}}$, our main RelBERT model has 354M parameters. This relatively small size makes RelBERT convenient and efficient to use in prac-

tice. Surprisingly, we found RelBERT to significantly outperform language models which are several orders of magnitude larger.

While many NLP tasks can now be solved by prompting large LMs, learning explicit representations remains important for tasks that require transparency or efficiency. For instance, we envision that RelBERT can play an important role in the context of semantic search, e.g. to find relevant context for retrieval augmented LMs (Guu et al., 2020). Explicit representations also matter for tasks that cannot easily be described using natural language instructions, such as ontology alignment (He et al., 2022) and completion (Chen et al., 2022b; Li et al., 2019a), where relation embeddings should intuitively also be clearly useful. More generally, RelBERT has the potential to improve applications that currently rely on commonsense KGs such as ConceptNet, e.g. commonsense question answering with smaller LMs (Yasunaga et al., 2021) and scene graph generation (Chen et al., 2023).

5 Modelling Graded Relations between Named Entities

5.1 Introduction

We have been focusing on modelling word analogy by LMs, and found that learning relational similarity enables LMs to distil the relational knowledge in Chapter 4. The resulting models, RelBERT, achieves SoTA in analogy benchmark, outperforming other large LMs. Nevertheless, relations are often a matter of degree (Rosch, 1975; Turney, 2006; Vulić et al., 2017) in practice rather than a classification like the analogy question. For instance, suppose we are interested in modelling whether one entity has been *influenced by* another one. While we could argue that most contemporary pop music has been influenced by the Beatles, in one way or another, clearly there are some bands that have been influenced more directly than others. Graded relations such as *influenced by*, *a competitor of* or *similar to* are typically not found in traditional KGs, while they can nonetheless be of central importance to applications. For instance, in the context of financial NLP, we may need to know

which companies are leaders and which are followers in a given field, who is competing with whom, and what strategic alliances exist. As another example, music recommendation systems often suggest artists based on the user's listening history, but these suggestions would be more helpful if the system could identify artists that have influenced or were influenced by artists the user already likes, as opposed to merely identifying similar artists. Studying how such relations can be modelled is thus clearly an important but under-explored research problem.

The subjective nature of graded relations makes it difficult to include them in traditional KGs. Moreover, for many of these relations, it would simply not be feasible to list all the (graded) instances in a comprehensive way. Taking inspiration from existing work on extracting KGs from large LMs, we therefore ask the following question: *are current large LMs capable of modelling graded relations between named entities in a meaningful way?* The task of modelling graded relations offers a number of unique challenges for large LMs. First, since this is essentially a ranking task, rather than a labelling task, designing suitable prompts is not straightforward. Second, the task requires making very fine-grained distinctions. For instance, while we can say that *Microsoft is known for Windows* and *Apple is known for MacOS*, the former statement represents a more prototypical instance of the *known for* relation, as Apple is perhaps best known for its hardware products (e.g. iPhone). It is currently unclear to what extent large LMs are able to capture such subtle differences. Finally, modelling graded relations requires comparing entities of different types. For instance, the *known for* relation has instances such as (*Microsoft*,*Windows*), (*the Beatles*, *Hey Jude*) and even (*France*,*wine*). Comparing instances of such a diverse nature poses a particular challenge, as such comparisons are almost never expressed in text.

In this chapter, we introduce RELENTLESS[1], a new dataset aimed at furthering the study of graded relations between named entities, where the dataset is available at https://huggingface.co/datasets/cardiffnlp/relentless. Our dataset covers five common graded relations: competitor/rival of, friend/ally of, influenced by, known for, and similar to. We then evaluate the ability of large LMs to rank entity pairs according to how much they satisfy these relations, given a description of the relation and five prototypical examples. Analysing the performance of several recent large LMs, including GPT-4, we find the best models to achieve a Spearman rank correlation of around 0.5 to 0.6. This shows that recent large LMs capture fine-grained relational knowledge to a meaningful extent, while at the same time still leaving a significant gap with human performance. For the open-source large LMs, we find that while the largest models achieve strong results, smaller models fail to outperform a naive baseline based on fastText vectors. GPT-3 performs well, albeit slightly below the best variants of Flan-T5 and OPT. Finally, we found ChatGPT and GPT-4 hard to use for this task, since the OpenAI API[2] does not allow us to compute its perplexity scores. As a result, we were not able to outperform GPT-3 with these models.

5.2 DATASET

We focus on the five relations which are shown in Table 5.2. These relations were chosen because of their graded character and because they can apply to a broad range of entities. We created a dataset with annotated entity pairs for each of the

[1]The name RELENTLESS refers to Relations between Entities, where Less refers to the idea of ordering.

[2]https://openai.com/blog/openai-api

5: This is clearly a positive example, and I would expect everyone to agree with this view.
4: I consider this to be a positive example, but I would not be surprised if some knowledgeable people consider this word pair to be borderline.
3: I consider this to be a borderline case: I find it hard to decide whether this is a positive or a negative example.
2: I consider this to be a negative example, but I would not be surprised if some knowledgeable people consider this word pair to be borderline.
1: This is clearly a negative example, and I would expect everyone to agree with this view.

Table 5.1: Rating scale for the 2nd annotation phase.

relations in three phases. Seven annotators were involved: four undergraduate students, one PhD student and two faculty members. The annotators were diverse in terms of age, gender, ethnicity and nationality. The students were recruited through an internal student employment service and were offered a remuneration of around £16 per hour. The total annotation effort was around 150 hours. The annotation process was split into three phases.

5.2.1 First phase

In the first phase, the annotators were asked to provide 15 entity pairs for each of the five relations. Specifically, the aim was to provide 5 prototypical examples (i.e. entity pairs that clearly satisfy the relationship), 5 borderline positive pairs, which only satisfy the relationship to some extent, and 5 borderline negative pairs, which do not satisfy the intended relationship but are nonetheless related in a similar way. After removing duplicates, this resulted in an average of 114 entity pairs for each relation, and 573 pairs in total. We augmented these entity pairs with a number of randomly chosen entity pairs. The entities for these random pairs were selected

Relation Type	Val	Test	Prototypical Examples	Middle Rank Examples
competitor/rival of	20	89	Dell : HP, Sprite : 7 Up, Israel : Palestine, Liverpool FC : Manchester United, Microsoft Teams : Slack	Macallan : Suntory, Marvel Comics : D.C. Comics, Borussia Dortmund : PSG, UK : France, Doctor Who : Game of Thrones
friend/ally of	20	92	Australia : New Zealand, Aznar : Bush, Extinction Rebellion : Greta Thunberg, Elsa : Anna, CIA : MI6	Kylo Ren : Rey, UK : Commonwealth, Darth Vader : Emperor Palpatine, The Beatles : Queen, Mark Drakeford : Rishi Sunak
influenced by	20	93	Europe : European Union, Plato : Socrates, Ethereum : Bitcoin, Messi : Maradona, Impressionism : Edouard Manet	Mike Tyson : Muhammad Ali, US : NASA, Acer : Asus, Vincent van Gogh : Bipolar disorder, Conservative Party : Labour Party
known for	20	108	Russell Crowe : Gladiator, Cadbury : chocolate, Paris : Eiffel Tower, Leonardo Da Vinci : Mona Lisa, Apple : iPhone	New Zealand : sheep, Le Corbusier : purism art, Sean Connery : Finding Forrester, Qualcomm : smartphones, Nikola Tesla : robotics
similar to	20	93	Coca-Cola : Pepsi, Ligue 1 : Bundesliga, Australia : New Zealand, The Avengers : The Justice League, Tesco : Sainsburys	NATO : United Nations, Iraq : Iran, cement : concrete, Cornwall : Brittany, Adele : Ed Sheeran

Table 5.2: Overview of the considered relations, showing the numbers of entity pairs in the validation and test sets, the five prototypical training examples, and five examples from the middle of the ranking of the entity pairs in the validation set.

from the 50,000 most popular Wikidata entities, in terms of the number of page views of the associated Wikipedia article.

	A	B	C	D	E	F	G	Others
A	100	62	81	71	75	75	75	84
B	62	100	61	57	62	57	60	66
C	81	61	100	73	72	74	75	84
D	71	57	73	100	67	67	70	77
E	75	62	72	67	100	70	72	77
F	75	57	74	67	70	100	69	76
G	75	60	75	70	72	69	100	79
Avarage	77	66	77	72	74	73	74	77

Table 5.3: Spearman correlation (%) between each pair of annotators (A,...,G), and between each annotator and the average score provided by the other six averaged over all the five relation types **after the 3rd and final quality enhancement annotation round.**

	A	B	C	D	E	F	G	Others
A	100	53	77	63	64	68	67	80
B	53	100	52	43	47	46	48	56
C	77	52	100	63	58	67	68	79
D	63	43	63	100	48	54	59	66
E	64	47	58	48	100	57	59	65
F	68	46	67	54	57	100	62	70
G	67	48	68	59	59	62	100	73
Average	70	55	69	61	62	65	66	70

Table 5.4: Spearman correlation (%) between each pair of annotators (A,...,G), and between each annotator and the average score provided by the other six averaged over all the five relation types **before the 3rd and final quality enhancement annotation round.**

5.2.2 Second phase

In the second phase, each annotator scored all the entity pairs provided by the annotators in phase 1, using the 5-point scale shown in Table 5.1. For this phase, annotators were encouraged to consult web sources (e.g. search engines such as Google) for a limited time in order to familiarize themselves with the considered entities, if needed. This was the most time-consuming annotation phase, taking al-

most 10 hours on average per annotator to complete.

5.2.3 Third phase

The third and final phase was aimed at resolving disagreements between the annotations from the second phase. Specifically, for each entity pair where there was a difference of 3 points between the highest and the lowest score, the annotator(s) with a diverging view were asked to check their previous annotation, and to either update their score or to provide a justification. A total of 255 unique entity pairs were checked in this way (310 scores were checked in total). We subsequently checked the justifications that were provided. In 13 cases, the justifications suggested that the other annotators might have missed some salient point. For these cases, the annotators with the opposite view were asked to re-check their previous annotation.

The final rankings for each relation were obtained by averaging the scores of the 7 annotators. Table 5.3 summarises the agreement between the annotators in terms of Spearman's rank correlation. The table shows the correlation between the individual annotators, as well as the correlation between each annotator and the average of the scores from the six other annotators. For comparison, Table 5.4 shows the agreement for the rankings that were obtained after phase 2, i.e. before the reconciliation step in the third phase. We can see that this reconciliation step improved the average agreement over all the annotators from 70 to 77.

We split the annotated entity pairs as follows. First, we selected a small training set consisting of five prototypical pairs for each relation. These entity pairs in training set could be used, for instance, for few-shot prompting strategies. The entity

pairs were selected (i) to be among the top-ranked entity pairs and (ii) to be sufficiently diverse (i.e. including entities of different types). Next, for each relation, we randomly selected 20 of the remaining entity pairs to be used as a validation set. This validation set was not used in our experiments, apart from the qualitative analysis. However, we release it so it can be used for further testing and experimentation without the risk of overfitting on the test set. The remaining entity pairs constitute the test set. Table 5.2 shows the prototypical entity pairs that were selected for each relation, as well as five examples of entity pairs from the validation set. The latter were selected from the middle of the ranking, typically with an average score of 3 to 4. We use the Spearman rank correlation between the predicted ranking and the ground truth ranking as the evaluation metric.

5.3 Baselines

In this section, we consider a number of baseline methods to predict the extent to which a given word pair satisfies a graded relation.

5.3.1 Human Performance

As a proxy of human performance, we report the average Spearman rank correlation between each annotator and the average of the other six in the test set, referred as *Human Upperbound*. Please note that these scores may differ from those presented in Table 5.3, as those agreement scores were computed on the entire dataset, not only the test set.

5.3.2 Embedding Models

As baselines of lexical relation embedding models, we consider fastText, a static word embedding, and RelBERT, the SoTA relation embedding model we proposed in § 4.2.

5.3.2.1 Word Embedding

First, we consider the fastText embeddings that were trained on Common Crawl with subword information[3]. Inspired by the tradition of modelling word analogies using vector differences, we represent each entity pair by subtracting the fasttext embedding of the first entity from the embedding of the second entity. We refer to the resulting vector as the fasttext relation embedding. For a given relation, we score an entity pair by taking the maximum cosine similarity between its fasttext relation embedding and the fasttext relation embedding of the five prototypical examples. We use the maximum, rather than e.g. the average, due to the diverse nature of these prototypical examples. Empirically, we confirmed that indeed using the maximum leads to better results overall. We refer this approach as $\text{fastText}_{\text{pair}}$.

As a naive baseline, we also consider a variant in which an entity pair is scored by taking the cosine similarity between the word embeddings of the two entities. Note that this baseline ignores both the description of the relation and the prototypical examples. It is based on the idea that prototypical pairs often involve closely related entities. We refer to this approach as $\text{fastText}_{\text{word}}$.

[3] https://fasttext.cc/

5.3.2.2 RelBERT

We use $\text{RelBERT}_{\text{BASE}}$ and $\text{RelBERT}_{\text{LARGE}}$ that were initialised from $\text{RoBERTa}_{\text{BASE}}$[4] and from $\text{RoBERTa}_{\text{LARGE}}$[5] respectively. Specifically, for a given relation, we score each entity pair as the maximum cosine similarity between its RelBERT encoding and the RelBERT encoding of the five prototypical examples.

5.3.3 Language Models

To score entity pairs using LMs, we create a prompt from the description of the relation and the five prototypical examples. The score of the entity pair then corresponds to the perplexity of the prompt. We consider two prompt templates: a binary QA template similar to the instructions provided to Flan-T5 for the task, and a targeted List Completion (LC) template. Writing the five prototypical examples as $[A_i, B_i]_{i=1...5}$ and the target entity pair as $[C, D]$, the QA template has the following form:

> Answer the question by yes or no. We know that $[A_1, B_1], \ldots, [A_5, B_5]$ are examples of `<desc>`. Are $[C, D]$ `<desc>` as well?
> Yes

The LC template has the following form:

> Complete the following list with examples of `<desc>`

[4] https://huggingface.co/relbert/relbert-roberta-base
[5] https://huggingface.co/relbert/relbert-roberta-large

Model	Inst-FT	Model Size	Name on HuggingFace
OPT_{125M}		125M	`facebook/opt-125m`
OPT_{350M}		350M	`facebook/opt-350m`
$\text{OPT}_{\text{1.3B}}$		1.3B	`facebook/opt-1.3b`
$\text{OPT}_{\text{2.7B}}$		2.7B	`facebook/opt-2.7b`
$\text{OPT}_{\text{6.7B}}$		6.7B	`facebook/opt-6.7b`
OPT_{13B}		13B	`facebook/opt-13b`
OPT_{30B}		30B	`facebook/opt-30b`
OPT_{66B}		66B	`facebook/opt-66b`
$\text{OPT-IML}_{\text{1.3B}}$	✓	1.3B	`facebook/opt-iml-1.3b`
$\text{OPT-IML}_{\text{30B}}$	✓	30B	`facebook/opt-iml-30b`
$\text{OPT-IML}_{\text{M-1.3B}}$	✓	1.3B	`facebook/opt-iml-max-1.3b`
$\text{OPT-IML}_{\text{M-30B}}$	✓	30B	`facebook/opt-iml-max-30b`
T5_{SMALL}		60M	`t5-small`
T5_{BASE}		220M	`t5-base`
T5_{LARGE}		770M	`t5-large`
T5_{3B}		3B	`t5-3b`
T5_{11B}		11B	`t5-11b`
$\text{Flan-T5}_{\text{SMALL}}$	✓	60M	`google/flan-t5-small`
$\text{Flan-T5}_{\text{BASE}}$	✓	220M	`google/flan-t5-base`
$\text{Flan-T5}_{\text{LARGE}}$	✓	770M	`google/flan-t5-large`
$\text{Flan-T5}_{\text{XL}}$	✓	3B	`google/flan-t5-xl`
$\text{Flan-T5}_{\text{XXL}}$	✓	11B	`google/flan-t5-xxl`
$\text{Flan-UL2}_{\text{20B}}$	✓	20B	`google/flan-ul2`

Table 5.5: The LMs used in the paper and their corresponding alias on HuggingFace model hub.

$[A_1, B_1]$

:

$[A_5, B_5]$

$[C, D]$

In both templates, `<desc>` is the description of the relation, as follows:

- *Rival:* entities that are competitors or rivals
- *Ally:* entities that are friends or allies

- *Inf:* what has influenced different entities
- *Know:* what entities are known for
- *Sim:* entities that are similar

We use the following LMs: OPT, OPT-IML, T5, Flan-T5, and Flan-UL2, where the model weights are obtained via HuggingFace. A complete list of the models on huggingface we used can be found in Table 5.5. We also use GPT-3 via the OpenAI API. We use `davinci` model, which is the most powerful model within the available GPT-3 models from OpenAI. We compute the perplexity over the whole input text for OPT, OPT-IML and GPT-3, while we use the last line of the input text (i.e., "Yes" for the QA template and $[C, D]$ for the LC template) to compute the perplexity on the decoder for T5, Flan-T5, and Flan-UL2.

5.3.4 Conversational Models

Additionally, we test two conversational LMs: ChatGPT (`gpt-3.5-turbo`)[6] and GPT-4 (`gpt-4`)[7]. These models are only available through the OpenAI API. Unfortunately, for these models, the API does not allow us to obtain the log-likelihood of each token. Therefore, we instead use a prompt which asks to sort the list of entity pairs directly. Writing the list of target word pairs as $[C_i, D_i]_{i=1...n}$, the prompt that we used has the following form:

> Consider the following reference list of `<desc>`:

[6] https://openai.com/blog/chatgpt
[7] https://openai.com/research/gpt-4

Model	Model Size	Rival	Ally	Inf	Know	Sim	Average
Human Upperbound		75.9	78.0	70.5	82.0	80.2	77.3
$\text{fastText}_{\text{pair}}$	-	28.0	12.0	3.0	20.0	21.0	17.0
$\text{fastText}_{\text{word}}$	-	25.0	10.0	7.0	24.0	20.0	17.0
$\text{RelBERT}_{\text{BASE}}$	110M	58.0	15.0	30.0	24.0	28.0	31.0
$\text{RelBERT}_{\text{LARGE}}$	335M	64.0	20.0	20.0	44.0	53.0	40.0

Table 5.6: Spearman's rank correlation (%) on the test set for the embedding models. Model size is measured as the number of parameters.

$[A_1, B_1]$

:

$[A_5, B_5]$

Now sort the entity pairs from the following list based on the extent to which they also represent `<desc>` in descending order. Do not include the pairs from the reference list. The output should contain all the entity pairs from the following list and no duplicates:

$[C_1, D_1]$

:

$[C_n, D_n]$

These conversational models often omit entity pairs from the output, especially those with lower similarity to the reference pairs. To deal with this, we simply concatenate those removed pairs to the bottom of the sorted output list.

5.4 Results

Table 5.6, Table 5.7, and Table 5.8 summarise the results of the comparison models. The best result is achieved by Flan-T5_{XXL} with the QA template, which scores

	Model	Model Size	Rival	Ally	Inf	Know	Sim	Average
T5	$T5_{SMALL}$	60M	10.0	-13.0	17.0	-6.0	8.0	3.0
	$T5_{BASE}$	220M	15.0	-7.0	6.0	-12.0	14.0	3.0
	$T5_{LARGE}$	770M	-3.0	4.0	-12.0	-19.0	-1.0	-6.0
	$T5_{XL}$	3B	-2.0	12.0	-8.0	17.0	-14.0	1.0
	$T5_{XXL}$	11B	7.0	1.0	-1.0	11.0	-4.0	3.0
	Flan-$T5_{SMALL}$	60M	31.0	-0.0	21.0	-3.0	8.0	11.0
	Flan-$T5_{BASE}$	220M	41.0	28.0	46.0	17.0	22.0	31.0
	Flan-$T5_{LARGE}$	770M	67.0	39.0	24.0	49.0	56.0	47.0
	Flan-$T5_{XL}$	3B	75.0	44.0	44.0	61.0	63.0	57.0
	Flan-$T5_{XXL}$	11B	74.0	**56.0**	44.0	**70.0**	**66.0**	**62.0**
	Flan-UL2	20B	**79.0**	51.0	47.0	67.0	57.0	60.0
OPT	OPT_{125M}	125M	35.0	31.0	46.0	10.0	9.0	26.0
	OPT_{350M}	350M	38.0	35.0	37.0	21.0	19.0	30.0
	$OPT_{1.3B}$	1.3B	44.0	33.0	46.0	29.0	31.0	37.0
	$OPT_{2.7B}$	2.7B	54.0	32.0	50.0	38.0	32.0	41.0
	$OPT_{6.7B}$	6.7B	53.0	33.0	39.0	46.0	34.0	41.0
	OPT_{13B}	13B	63.0	39.0	43.0	61.0	43.0	50.0
	OPT_{30B}	30B	61.0	38.0	48.0	62.0	45.0	51.0
	OPT-$IML_{1.3B}$	1.3B	45.0	27.0	42.0	21.0	26.0	32.0
	OPT-IML_{30B}	30B	57.0	37.0	36.0	53.0	35.0	44.0
	OPT-$IML_{M\text{-}1.3B}$	1.3B	42.0	25.0	38.0	16.0	29.0	30.0
	OPT-$IML_{M\text{-}30B}$	30B	58.0	36.0	39.0	43.0	42.0	43.0
	GPT-$3_{davinci}$*	175B	67.0	35.0	**50.0**	61.0	35.0	50.0

Table 5.7: Spearman's rank correlation (%) on the test set for the LMs with the QA template grouped by the model family. The best correlation in each relation type is highlighted by bold characters. Model size is measured as the number of parameters. Models marked with * are not openly available.

62.0%. In general, the performance of this model remains far below the performance upper bound suggested by the inter-annotator agreement (80%). Surprisingly, however, for the *rival of* relation, the human upper bound is outperformed by Flan-UL2. In contrast, the *friend/ally of* relation appears to be particularly challenging. Among the LM methods, the LC template generally leads to the best results, with the exception of Flan-T5 and Flan-UL2. This is not entirely surprising given that Flan models have been fine-tuned using instructions similar to the QA template (see § 5.3.3). Beyond the encoder-decoder LMs, OPT_{13B} and GPT-$3_{davinci}$

	Model	Model Size	Rival	Ally	Inf	Know	Sim	Average
T5	$T5_{SMALL}$	60M	20.0	33.0	24.0	11.0	10.0	19.0
	$T5_{BASE}$	220M	35.0	35.0	38.0	20.0	13.0	28.0
	$T5_{LARGE}$	770M	29.0	8.0	26.0	11.0	22.0	19.0
	$T5_{XL}$	3B	47.0	28.0	50.0	33.0	26.0	37.0
	$T5_{XXL}$	11B	33.0	8.0	24.0	18.0	15.0	19.0
	Flan-$T5_{SMALL}$	60M	38.0	33.0	24.0	16.0	7.0	24.0
	Flan-$T5_{BASE}$	220M	36.0	31.0	28.0	17.0	-0.0	22.0
	Flan-$T5_{LARGE}$	770M	41.0	19.0	36.0	24.0	22.0	29.0
	Flan-$T5_{XL}$	3B	40.0	17.0	35.0	27.0	31.0	30.0
	Flan-$T5_{XXL}$	11B	61.0	32.0	47.0	44.0	40.0	45.0
	Flan-UL2	20B	60.0	28.0	49.0	53.0	37.0	45.0
OPT	OPT_{125M}	125M	41.0	37.0	51.0	23.0	13.0	33.0
	OPT_{350M}	300M	41.0	33.0	47.0	36.0	18.0	35.0
	$OPT_{1.3B}$	1.3B	58.0	39.0	54.0	45.0	42.0	48.0
	$OPT_{2.7B}$	2.7B	65.0	41.0	58.0	56.0	42.0	52.0
	$OPT_{6.7B}$	6.7B	71.0	**42.0**	59.0	61.0	47.0	56.0
	OPT_{13B}	13B	72.0	41.0	55.0	70.0	**55.0**	**59.0**
	OPT_{30B}	30B	71.0	39.0	57.0	69.0	53.0	58.0
	OPT-$IML_{1.3B}$	1.3B	57.0	39.0	56.0	51.0	35.0	47.0
	OPT-IML_{30B}	30B	65.0	36.0	55.0	70.0	47.0	55.0
	OPT-$IML_{M\text{-}1.3B}$	1.3B	55.0	37.0	57.0	49.0	33.0	46.0
	OPT-$IML_{M\text{-}30B}$	30B	62.0	36.0	57.0	67.0	46.0	53.0
	GPT-$3_{davinci}$*	175B	72.0	39.0	**64.0**	**73.0**	47.0	**59.0**

Table 5.8: Spearman's rank correlation (%) on the test set for the LMs with the LC template grouped by the model family. The best correlation in each relation type is highlighted by bold characters. Model size is measured as the number of parameters. Models marked with * are not openly available.

perform the best, even outperforming the instruction fine-tuned OPTs (OPT-IML and OPT-IML_{MAX}). GPT-$3_{davinci}$ is the best model in the *influenced by* and *known for* relations. Although Flan-$T5_{XXL}$ and Flan-UL2 are the top-2 on average, they perform poorly on the *influenced by* relation, underperforming GPT-$3_{davinci}$ and OPT_{13B} by a great margin.

Among the embedding based models, fastText generally performs poorly. The performance of $RelBERT_{LARGE}$ is remarkably strong, considering that this is a concept-

	ChatGPT	GPT-4
Rival	-0.9 (0.0%)	62.5 (100.0%)
Ally	42.5 (56.8%)	55.8 (100.0%)
Inf	17.5 (91.1%)	35.9 (94.4%)
Know	15.5 (86.7%)	60.8 (100.0%)
Sim	14.7 (80.9%)	69.3 (98.9%)
Average	17.9 (63.1%)	56.9 (98.7%)

Table 5.9: Spearman's rank correlation (%) on the test set for conversational LMs with the percentage of word pairs included in the output.

based relation model that was not trained on relations between named entities.

As far as the OpenAI conversational models are concerned, we can see that GPT-4 achieves the best result on the *similar to* relation. The poor performance of ChatGPT suggests that the considered list ranking prompt may be hard to understand for this model, or that the task of ranking around 100 pairs may in itself be too complicated.

5.4.1 Conversational Models

Table 5.9 shows the results and percentage of retrieved pairs of the conversational LMs. We can see that GPT-4 significantly improves on ChatGPT. However, neither model outperforms GPT-3 with the LC template. This suggests that the considered list ranking prompt may be hard to understand or confusing for these models, or that the task of ranking around 100 pairs is excessively long for the current capabilities of these models. We also observed that ChatGPT tends to omit more pairs from its output than GPT-4.

5.5 Analysis

In this section, we analyse different aspects of the model performance in order to gain a better understanding of the behaviour of LMs. First, we perform an analysis on the effect size (§ 5.5.1). Then, we experiment with different zero-shot and few-shot learning set-ups (§ 5.5.2), and we present a qualitative analysis of the predictions (§ 5.5.3). For the latter two analyses, we focus on the best performing models for each LM family from the main experiment, using their optimal prompts: Flan-UL2, Flan-$T5_{XXL}$, OPT_{13B}, and GPT-$3_{davinci}$. Note that we omit Flan-UL2 from the model size analysis as Flan-UL2 only has a single size.

5.5.1 Model Size

In this section, we analyse the effect of model size. Figure 5.1 and Figure 5.2 visualise the performance of the different model families in function of model size. For Flan-T5, OPT, and OPT-IML we can see a strong correlation between performance and size. Nevertheless, the result of the largest OPT models suggests that a plateau in performance may have been reached at 13B. Moreover, for T5 we do not see an improvement in performance for larger models.

5.5.2 Zero-shot/Few-shot Learning

In the main experiments, for each relation, models had access to a description as well as five prototypical examples. To analyse the impact of these five examples, we

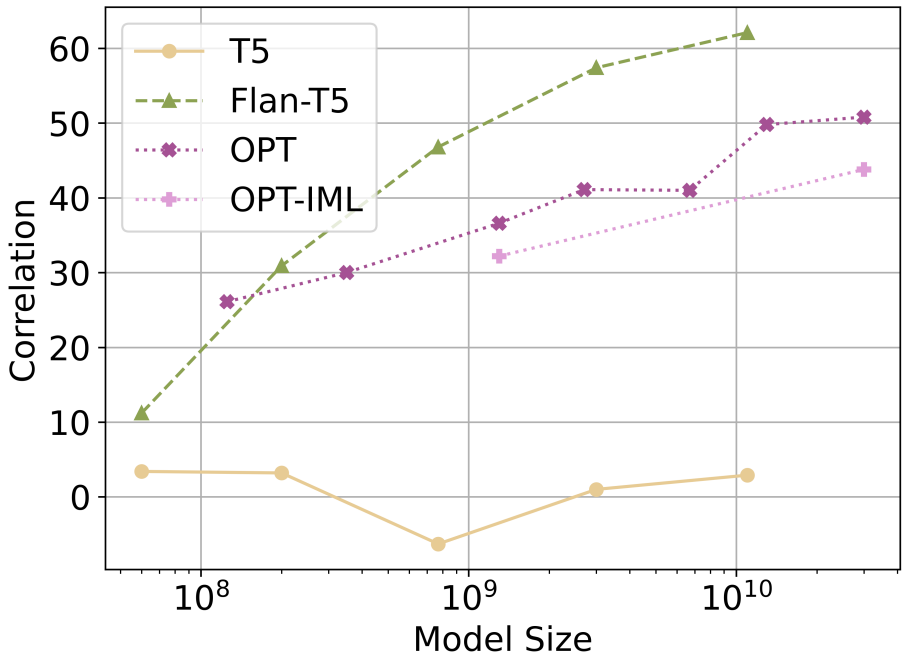

Figure 5.1: The average Spearman's rank correlation results among the five relation types along with the model size for the QA template.

now describe experiments in which only the description is provided (i.e. zero-shot) as well as experiments where only 1 or 3 examples are given (few-shot). For the few-shot setting, we use the same QA and LC templates as in the main experiment. For the 3-shot experiments, we randomly choose 3 of the 5 templates; and similar for the 1-shot experiments. Since this introduces some randomness, we report results for three different samples. For the zero-shot setting, we modify the templates as follows. The QA template now becomes:

> Answer the question by yes or no. Are $[C, D]$ `<desc>`?
>
> Yes

while the zero-shot LC template has the following form:

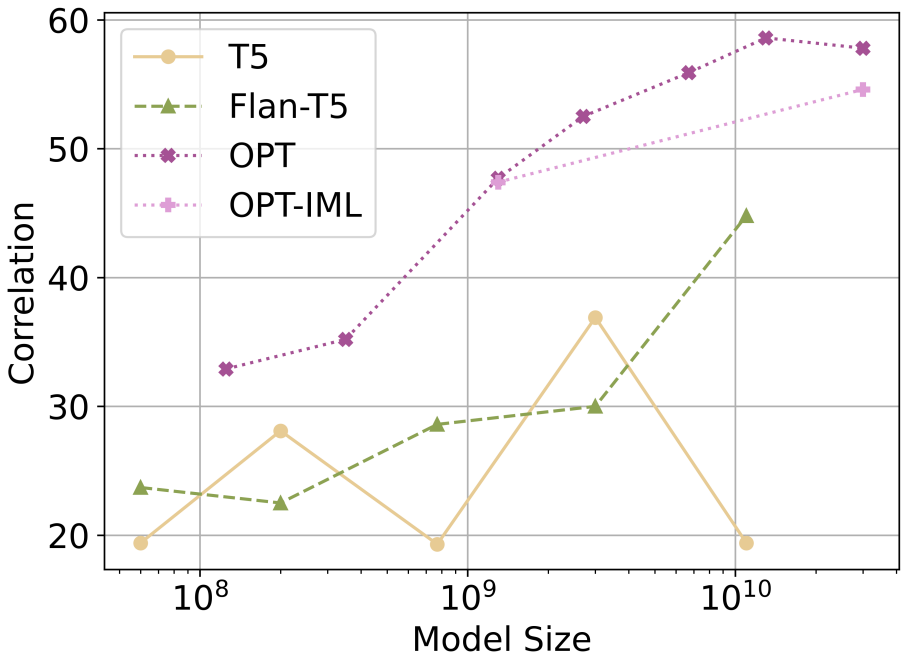

Figure 5.2: The average Spearman's rank correlation results among the five relation types along with the model size for the LC template.

> Complete the following list with examples of `<desc>`?
>
> $[C, D]$

Figure 5.3 shows the results for the QA template. We can see that all models improve when more prototypical examples are provided, with the zero-shot performance of Flan-UL2 being an outlier. Remarkably, Flan-UL2 achieves 62.5% accuracy in the zero-shot setting, which is competitive with the 5-shot results in Table 5.7. Flan-$T5_{XXL}$ also achieves a zero-shot result of 54.5%, which is better than most of the models in the main experiments (for the 5-shot setting). In the zero-shot setting, OPT_{13B} performs better than GPT-$3_{davinci}$, but GPT-$3_{davinci}$ quickly improves as more examples are provided, clearly outperforming OPT_{13B} in the 5-shot setting. Figure 5.4 shows the results for the LC template. We can again see

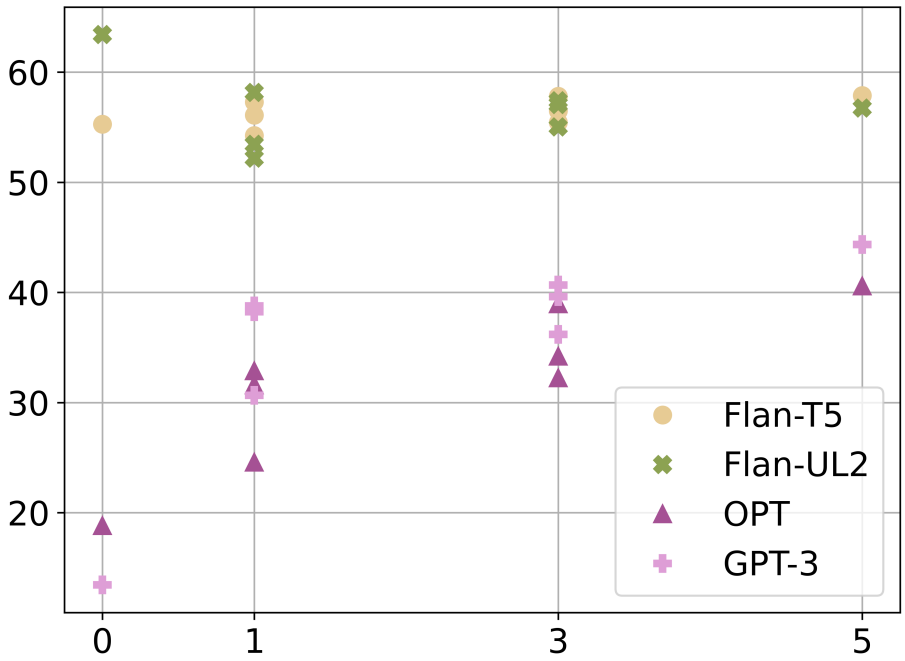

Figure 5.3: Spearman's rank correlation averaged over the five relation types with different number of the prototypical examples for QA template. For 1-shot and 3-shot examples, we report the each correlation of the three individual runs.

that providing more examples consistently benefits all models. Unlike for the QA template, however, Flan-T5$_{\text{XXL}}$ performs poorly in the zero-shot setting. Moreover, OPT$_{\text{13B}}$ now sees the largest improvement between the zero-shot and 5-shot settings.

5.5.3 Qualitative Analysis

To better understand the predictions of the models, we analyse the most flagrant mistakes. Specifically, we focus on those entity pairs whose predicted rank is in the top 30%, while being in the bottom 30% of the gold ranking, and vice versa. Table 5.10 shows the entity pairs from the validation set for which this was the case.

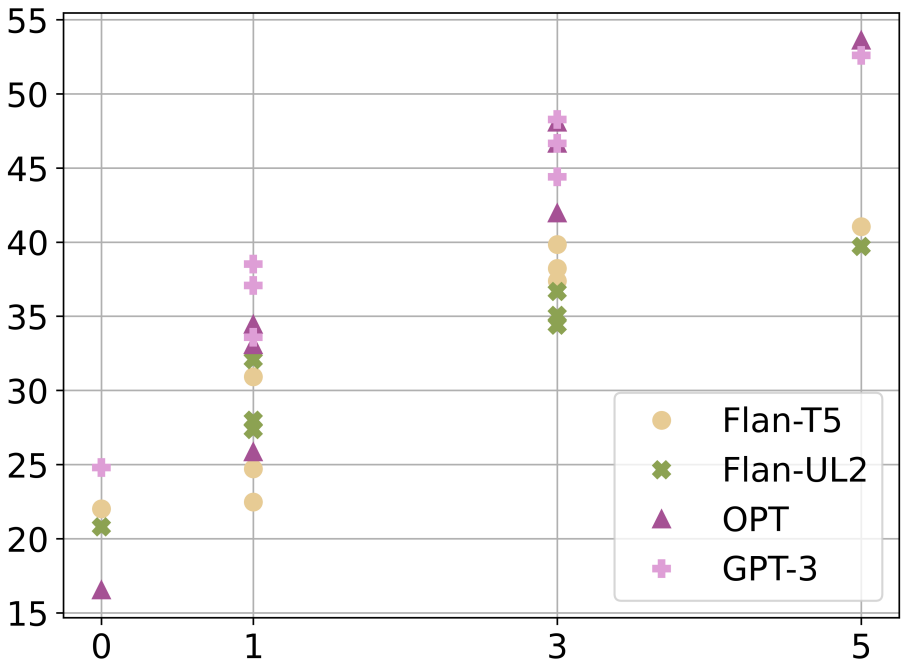

Figure 5.4: Spearman's rank correlation averaged over the five relation types with different number of the prototypical examples for LC template. For 1-shot and 3-shot examples, we report the each correlation of the three individual runs.

For this analysis, we look at the models using their optimal templates: i.e., Flan-T5 with the QA template, and the other models with the LC template.

When looking at the instances that mistakenly end up in the top 30%, we see entities which are closely related in some way (e.g. "Darth Vader : Obi-Wan Kenobi) while not actually satisfying the intended relation. For the "Rihanna : Stevie Wonder" example, it seems that the models were confused by the apparent resemblance with "Messi : Maradona", one of the prototypical examples of the influenced-by relation: both cases involve well-known entities of the same semantic type (i.e. musicians and footballers respectively), where only the first entity is a contemporary celebrity. When looking at the examples of the bottom 30%, we can see entities which only recently became prominent (e.g. OpenAI and Liz Truss), high-

		Incorrectly predicted to be in the top 30%	Incorrectly predicted to be in the bottom 30%
Flan-T5$_{\text{XXL}}$	Ally	Darth Vader:Obi-Wan Kenobi	Thomas Carlyle:Charles Dickens, Maximus Decimus Meridius:Juba
	Inf		Vape:cigarette
Flan-UL2	Ally		Owain Glyndwr:Charles VI of France, Maximus Decimus Meridius:Juba
	Inf	Rihanna:Stevie Wonder	
	Know	Qualcomm:smartphones	
OPT$_{\text{13B}}$	Rival	Betty White:Charli D'Amelio	Wickes:ScrewFix
	Ally	Darth Vader:Obi-Wan Kenobi, Schindler's List:Baz Luhrmann	Thomas Carlyle:Charles Dickens, Microsoft:OpenAI
	Inf	Rihanna:Stevie Wonder	UK:Winston Churchill, Vape:cigarette
	Sim	Domino's:YO! Sushi	
GPT-3$_{\text{davinci}}$	Rival		Wickes:ScrewFix
	Ally	Darth Vader:Obi-Wan Kenobi	
	Inf	Rihanna:Stevie Wonder	Vape:cigarette, Liz Truss:Thatcher

Table 5.10: Validation examples of incorrect predictions made by the three best models in the top 30% or bottom 30%.

lighting the limitation of using LMs that have not been trained on the most recent data. The "UK : Winston Churchill" example illustrates how the models can struggle with cases involving entities of different semantic types. Finally, the "Wickes : ScrewFix" and "EE: Three UK" examples require knowledge of the UK market, which the models may fail to grasp without further context.

5.6 Summary

We have proposed the task of modelling graded relations between named entities, with a new dataset. The task consists in ranking entity pairs according to how much they satisfy a given graded relation, where models only have access to the description of the relation and five prototypical instances per relation. To assess the difficulty of the task, we analysed a large number of baselines, including public large LMs of up to 30B parameters, state-of-the-art relation embedding models, and closed large LMs such as GPT-4. We found significant performance differences between the largest LMs and their smaller siblings, which highlights the progress achieved in NLP in the last few years by scaling up LMs. However, even the largest models trail human performance by around 15 percentage points.

Given the result we have obtained from the previous chapter chapter 4, it is interesting to see RelBERT struggling to solve the task, while it can achieve SoTA performance in most of the analogy questions. From this perspective, the relation embedding models still have a room of improvement, and it opens up a new direction of the future works that improves the generalization ability of relation embedding models in the named entity relation not appeared in the training dataset.

6 Conclusions and Future Work

6.1 Conclusion

This thesis aims to achieve the goal of understanding relational knowledge in LMs and developing a method to extract such relational knowledge from LM to apply in the downstream tasks with relation understanding. The area of analyzing LM for better understanding of its various capability gains popularity in recent few years, yet the relational knowledge in LMs had been paid less attention, and understudied in the literature. Our study provides a suite of evaluation protocol to analyze the relation knowledge of LMs to shed light on the ability of LMs to capture the relational knowledge. The proposed datasets include the task of analogy question from various domain including educational settings, commonsense KG, named entities, and scientific metaphor, and we have shown that with scoring functions tailored for solving analogy questions, LMs can outperform the other statistical baselines, but not all LMs are able to achieve a meaningful improvement over word embeddings.

Moreover, we have proposed a framework to distil the relational knowledge stored

in LM to achieve a relation embedding model that can achieve SoTA performance on analogy questions. The proposed framework fine-tunes LM to achieve lexical relation embedding model, *RelBERT*, i.e. vector representations of pairs of words which capture their relationship. Most importantly, we found that RelBERT is capable of modelling relationships that are not covered by the training data. We further extend our study to measure the capability of LM to understand relational knowledge by proposing new dataset that focuses on the relationship among named entities. The newly proposed task consists in ranking entity pairs according to how much they satisfy a given graded relation, where models only have access to the description of the relation and five prototypical instances per relation. The dataset turns out to be a challenging dataset even for the SoTA relation embedding models, that suggests a room of another future research direction to improve the relational understanding of the LMs.

6.2 Future Work

In this section, we describe future research directions to the studies we have done in this thesis.

6.2.1 Multilingual Relational Knowledge

The representation learning with multilinguality is an essential study in NLP. Many tasks have been extended to non-English language, such as QA (Artetxe et al., 2020; Lewis et al., 2020b; Clark et al., 2020), NER (Pan et al., 2017), dependency parsing (Nivre et al., 2018), paraphrase (Yang et al., 2019), and Natural Language

Inferrence (NLI) (Conneau et al., 2018). Also, many multilingual LMs are proposed in recent few years (Conneau et al., 2019; Liang et al., 2023; Scao et al., 2022; Xue et al., 2021; Liu et al., 2020), that are pre-trained over a large multilingual corpus. In spite of such trend in NLP, the relational knowledge understanding remains in English, and our studies in this thesis are limited to English either. Thus, we are lack of mutilingual study on the relational knowledge understanding, which is a natural line of future work.

6.2.2 Improving Methods for RelEntLess

In Chapter 5, we have introduced RelEntLess, a dataset for graded relation modelling between named entities, and shown that it was a challenging task for either of the large LMs or the SoTA relation embedding model. In this thesis, we focus on understanding the task and analyze the baseline, rather than achieving SoTA on the task. Subsequently, there is a room of improvement by investigating the baselines. For example, the prompt we used for LM baseline is underexplored. Thus, to improve the baselines will be a future work in understanding graded relation understanding.

6.2.3 Application of RelBERT in Downstream Tasks

In Chapter 4, we introduced RelBERT, the SoTA relation embedding model on the analogy question benchmarks. However, the application of RelBERT are limited to analogy questions and lexical classification in this thesis. Potentially, RelBERT can be applied for relation extraction and KG completion, but it requires some modifi-

cations or adaptation of current RelBERT, that will be an important future work.

6.2.4 Scaling Up RelBERT

Although RelBERT can outperform all the other large LMs to establish SoTA in analogy question benchamrk, on RelEntLess dataset, we find that RelBERT is worse than large LMs, while being competitive to the LMs in a same degree of parameter size. This suggests the possibility that RelBERT could improve its capability in a challenging task such as RelEntLess, if we could scale up the RelBERT with larger LMs. Due to the limited computational budget, we have not gone beyond the size of $\text{RoBERTa}_{\text{LARGE}}$, so the scaling up RelBERT is one possible future direction to extend the work in this thesis.

7 NLP Open Source Software

Recent success of NLP largely gain benefit from the open source software, which remove the burden of implementing the SoTA method in most case, and anyone can play with the SoTA models even without coding (Wolf et al., 2020; Abid et al., 2019). In the previous chapters, we have released the open source library related to the study in each work as below:

- `relbert`[1]: A Python package to extract a semantic embedding vector on a word pair with various RelBERT models we studied in chapter 4.
- `relentless` [2]: A dataset proposed in chapter 5 that we released public.

In this chapter, we introduce another four distinct NLP open source software that we developed to facilitate the use of NLP applications:

- `kex` https://pypi.org/project/kex/: A Python package for Keyword extraction (§ 7.1).

[1] https://pypi.org/project/relbert/
[2] https://huggingface.co/datasets/cardiffnlp/relentless

- `tner` https://pypi.org/project/tner/: A Python package and web application for NER fine-tuning/inference/evaluation (§ 7.2).
- `tweetnlp` https://pypi.org/project/tweetnlp/: A Python package and web interface for various set of NLP models on Twitter (§ 7.3).
- `lmqg` https://pypi.org/project/lmqg/: A Python package and web application for Question and Answer Generation (QAG) fine-tuning/inference/evaluation (§ 7.4).

7.1 KEX

Keyword extraction has been an essential task in many scientific fields as a first step to extract relevant terms from text corpora. Despite the simplicity of the task, it still poses practical problems, and often researchers resort to simple but reliable techniques such as Term Frequency-Inverse Document Frequency (TF-IDF) (Jones, 1972). In turn, term weighting schemes such as TF-IDF paved the way for developing large-scale Information Retrieval (IR) systems (Ramos et al., 2003; Wu et al., 2008). Its simple formulation is still widely used nowadays, not only for keyword extraction but also as an important component in IR (Jabri et al., 2018; Marcos-Pablos and García-Peñalvo, 2020) and NLP tasks (Riedel et al., 2017; Arroyo-Fernández et al., 2019).

While there exist supervised and neural techniques (Lahiri et al., 2017; Xiong et al., 2019; Sun et al., 2020), as well as ensembles of unsupervised methods (Campos et al., 2020; Tang et al., 2020) that can provide competitive performance, in this paper we go back to the basics and analyze in detail the single components of unsupervised methods for keyword extraction. In fact, it is still common to rely on unsupervised methods for keyword extraction given their versatility and the lack of training sets in specialized domains.

In order to fill this gap, in this section, we developped `kex`[3], a Python library that includes various types of unsupervised keyword extraction models. With the `kex` library, we perform an extensive analysis of single unsupervised keyword extraction techniques in a wide range of settings and datasets. To the best of our knowledge, this is the first large-scale empirical evaluation performed across base statistical and

[3] https://github.com/asahi417/kex

graphical keyword extraction methods. Our analysis sheds light on some properties of statistical methods largely unknown. For instance, our experiments show that a statistical weighting scheme based on the hypergeometric distribution such as lexical specificity (Lafon, 1980) can perform at least as well as or better than TF-IDF (Jones, 1972), while having additional advantages with respect to flexibility and efficiency. As for the graph-based methods, they can be more reliable than statistical methods without being considerably slower in practice. In fact, graph-based methods initialized with TF-IDF or lexical specificity performs best overall.

7.1.1 Keyword Extraction

Given a document with m words $[w_1 \cdots w_m]$, keyword extraction is a task to find n noun phrases, which can comprehensively represent the document. As each of such phrases consists of contiguous words in the document, the task can be seen as an ordinary ranking problem over all candidate phrases appeared in the document. A typical keyword extraction pipeline is thus implemented as, first, to construct a set of candidate phrases $\mathcal{P}_d$ for a target document d and, second, to compute importance scores for all of individual words in d[4]. Finally, the top-n phrases $\{y_j | j = 1 \dots n\} \subset \mathcal{P}_d$ in terms of the aggregated score are selected as the prediction (Mihalcea and Tarau, 2004). Figure 7.1 shows an overview of the overarching methodology for unsupervised keyword extraction.

To compute word-level scores, there are mainly two types of approach: statistical and graph-based. There are also contributions that focus on training supervised models for keyword extraction (Witten et al., 2005; Liu et al., 2010). However, due

[4]In the case of multi-token candidate phrases, this score is averaged among its tokens.

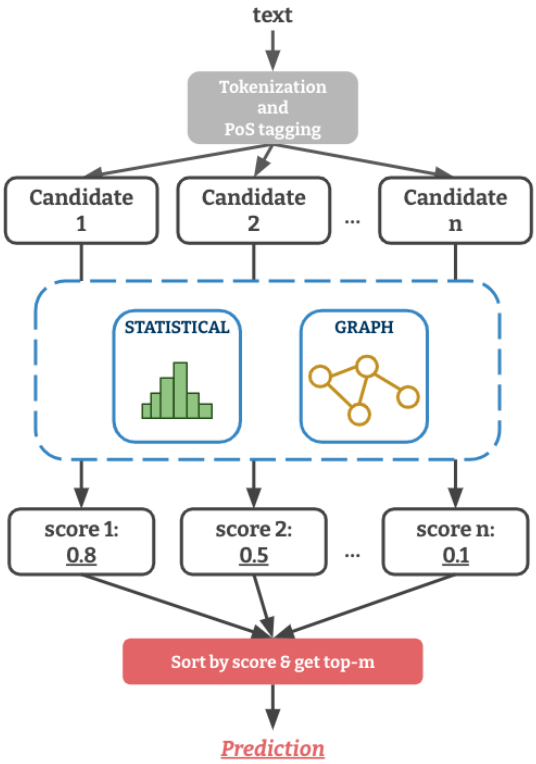

Figure 7.1: Overview of the keyword extraction pipeline.

to the absence of large labeled data and domain-specificity, most efforts are still unsupervised, which is the focus of this section.

7.1.1.1 Statistical Models

A statistical model attains an importance score based on word-level statistics or surface features, such as the word frequency or the length of word.[5] A simple keyword extraction method could be to simply use Term Frequency (TF) as a scoring

[5]The term *Statistical* may not be strictly accurate to refer to TF-IDF or purely frequency-based models, but in this case we follow previous conventions by grouping all these methods based on word-level frequency statistics as *statistical* (Aizawa, 2003).

function for each word, which tend to work reasonably well. However, this simple measure may miss important information such as the relative importance of a given word in a corpus. For instance, prepositions such as *in* or articles such as *the* tend to be highly frequent in a text corpus. However, they barely represent a keyword in a given text document. To this end, different variants have been proposed, which we summarize in two main alternatives: TF-IDF (§ 7.1.1.1) and *lexical specificity* (paragraph 7.1.1.1).

TF-IDF. TF-IDF (Jones, 1972) is one of most popular and effective methods used for statistical keyword extraction (El-Beltagy and Rafea, 2009), as well as still being an important component in modern information retrieval applications (Marcos-Pablos and García-Peñalvo, 2020; Guu et al., 2020). Given a set of documents $\mathcal{D}$ and a word w from a document $d \in \mathcal{D}$, TF-IDF is defined as the proportion between its word frequency and its inverse document frequency[6], as

$$s_{tfidf}(w|d) = tf(w|d) \cdot \log_2 \frac{|\mathcal{D}|}{df(w|\mathcal{D})} \tag{7.1}$$

where we define $|\cdot|$ as the number of elements in a set, $tf(w|d)$ as a frequency of w in d, and $df(w|\mathcal{D})$ as a document frequency of w over a dataset $\mathcal{D}$. In practice, $tf(w|d)$ is often computed by counting the number of times that w occurs in d, while $df(w|\mathcal{D})$ by the number of documents in $\mathcal{D}$ that contain w.

To give a few examples of statistical models based on TF-IDF and its derivatives in a keyword extraction context, KP-miner (El-Beltagy and Rafea, 2009) utilizes TF-IDF, a word length, and the absolute position of a word in a document to de-

[6]While there are other formulations and normalization techniques for TF-IDF (Paik, 2013), in this paper we focus on the traditional inverse-document frequency formulation.

termine the importance score, while RAKE (Rose et al., 2010) uses the term degree, the number of different word it co-occurs with, divided by TF. Recently, YAKE (Campos et al., 2020) established strong baselines on public datasets by combining various statistical features including casing, sentence position, term/sentence-frequency, and term-dispersion. In this paper, however, we focus on the vanilla implementation of TF and TF-IDF.

Lexical Specificity. Lexical specificity (Lafon, 1980) is a statistical metric to extract relevant words from a subcorpus using a larger corpus as reference. In short, lexical specificity extracts a set of most representative words for a given text based on the hypergeometric distribution. The hypergeometric distribution represents the discrete probability of k successes in n draws, without replacement. In the case of lexical specificity, k represents the word frequency and n the size of a corpus. While not as widely adopted as TF-IDF, lexical specificity has been used in similar term extraction tasks (Drouin, 2003), but also in textual data analysis (Lebart et al., 1998), domain-based disambiguation (Billami et al., 2014), or as a weighting scheme for building vector representations for concepts and entities (Camacho-Collados et al., 2016) or sense embeddings (Scarlini et al., 2020) in NLP. Formally, the lexical specificity for a word w in a document d is defined as

$$s_{spec}(w|d) = -\log_{10} \sum_{l=f}^{F} P_{hg}(x{=}l, m_d, M, f, F) \tag{7.2}$$

where m_d is the total number of words in d and $P_{hg}(x{=}l, m, M, f, F)$ represents the probability of a given word to appear l times exactly in d according to the hypergeometric distribution parameterised with m_d, M, f, and F, which are defined as

below.

$$M = \sum_{d \in \mathcal{D}} m_d, \; f = tf(w|d), \; F = \sum_{d \in \mathcal{D}} tf(w|d) \tag{7.3}$$

Note also that, unlike in TF-IDF, for lexical specificity a perfect partition of documents of $\mathcal{D}$ (reference corpus) is not required. This also opens up to other possibilities, such as using larger corpora as reference, for example.

7.1.1.2 Graph-based Methods

The basic idea behind graph-based methods is to identify the most relevant words from a graph constructed from a text document, where words are nodes and their connections are measured in different ways (Beliga et al., 2015). For this, PageRank (Page et al., 1999) and its derivatives have proved to be highly successful (Mihalcea and Tarau, 2004; Wan and Xiao, 2008a; Florescu and Caragea, 2017; Sterckx et al., 2015; Bougouin et al., 2013).

Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph where $\mathcal{V}$ and $\mathcal{E}$ are its associated set of vertices and edges. In a typical word graph construction on a document d (Mihalcea and Tarau, 2004), $\mathcal{V}$ is defined as the set of all unique words in d and each edge $e_{w_i,w_j} \in \mathcal{E}$ represents a strength of the connection between two words $w_i, w_j \in \mathcal{V}$. Then, a Markov chain from w_j to w_i on a word graph can be defined as

$$p(w_i|w_j) = (1-\lambda)\frac{e_{w_i,w_j}}{\sum_{w_k \in \mathcal{V}_i} e_{w_i,w_k}} + \lambda p_b(w_i) \tag{7.4}$$

where $\mathcal{V}_i \subset \mathcal{V}$ is a set of incoming nodes to w_i, $p_b(\cdot)$ is a prior probabilistic distribution over $\mathcal{V}$, and $0 \leq \lambda \leq 1$ is a parameter to control the effect of $p_b(\cdot)$. This probabilistic model (7.4) is commonly known as the random surfer model (Page et al.,

1999). The prior term $p_b(\cdot)$, which is originally a uniform distribution, is introduced to enable any transitions even if there are no direct connections among them. Once a word graph is built, PageRank is applied to estimate a probability $\hat{p}(w)$ for every word $w \in \mathcal{V}$, which is used as an importance score.

TextRank (Mihalcea and Tarau, 2004) uses an undirected graph and defines the edge weight as $e_{w_i,w_j} = 1$ if w_i and w_j co-occurred within l contiguous sequence of words in d, otherwise $e_{w_i,w_j} = 0$. SingleRank (Wan and Xiao, 2008a) extends TextRank by modifying the edge weight as the number of co-occurrence of w_i and w_j within the l-length sliding window and ExpandRank (Wan and Xiao, 2008b) multiplies the weight by cosine similarity of TF-IDF vector within neighbouring documents. To reflect a statistical prior knowledge to the estimation, recent works proposed to use non-uniform distributions for $p_b(\cdot)$. Florescu and Caragea (2017) observed that keywords are likely to occur very close to the first few sentences in a document in academic paper and proposed PositionRank in which $p_b(\cdot)$ is defined as the inverse of the absolute position of each word in a document. Topical PageRank (TPR) (Jardine and Teufel, 2014; Sterckx et al., 2015) introduces a topic distribution inferred by LDA as a $p_b(\cdot)$, so that the estimation contains more semantic diversity across topics. TopicRank (Bougouin et al., 2013) clusters the candidates before running PageRank to group similar words together, and MultipartiteRank (Boudin, 2018) extends it by employing a multipartite graph for a better candidate selection within a cluster.

Finally, there are a few other works that directly run graph clustering (Liu et al., 2009; Grineva et al., 2009), using edges to connect clusters instead of words, with semantic relatedness as a weight. Although these techniques can capture high-level semantics, the relatedness-based weights rely on external resources such as

Data	Size	Domain	Type	Divers.	# NPs		# tokens	
					avg	std	avg	std
KPCrowd	500	-	news	0.44	77	62.0	447	476.7
Inspec	2000	CS	abstract	0.55	27	12.3	138	66.6
Krapivin2009	2304	CS	article	0.12	815	252.1	9131	2524.4
Nguyen2007	209	-	article	0.15	618	113.9	5931	1023.1
PubMed	500	BM	article	0.18	566	196.5	4461	1626.4
Schutz2008	1231	BM	article	0.29	630	287.7	4201	2251.1
SemEval2010	243	CS	article	0.13	898	207.7	9740	2443.4
SemEval2017	493	-	paragraph	0.54	40	12.9	198	60.3
citeulike180	183	BI	article	0.21	822	173.0	5521	978.8
fao30	30	AG	article	0.21	774	93.2	5438	927.5
fao780	779	AG	article	0.19	776	147.2	5591	902.4
theses100	100	-	article	0.21	728	131.3	5397	958.4
kdd	755	CS	abstract	0.59	16	17.0	82	93.0
wiki20	20	CS	report	0.15	817	322.4	7146	3609.8
www	1330	CS	abstract	0.58	18	16.5	91	89.1

Table 7.1: Dataset statistics, where size refers to the number of documents; diversity refers to a measure of variety of vocabulary computed as the number of unique words divided by the total number of words; number of noun phrases refers to candidate phrases extracted by our pipeline; number of tokens is the size of the dataset. In terms of statistics, we show the average and the standard deviation.

Wikipedia (Grineva et al., 2009), and thus add another layer of complexity in terms of generalization. For these reasons, they are excluded from this study.

7.1.2 Evaluation

In this section, we evaluate the keyword extraction models implemented in `kex` on various benchmarks.

7.1.2.1 Experimental Setting

Datasets. To evaluate the keyword extraction methods, we consider 15 different public datasets in English.[7] Each entry in a dataset consists of a source document and a set of gold keyphrases, where the source document is processed through the pipeline described in the next paragraph and the gold keyphrase set is filtered to include only phrases which appear in its candidate set. Table 7.1 provides high-level statistics of each dataset, including length and number of keyphrases[8] (both average and standard deviation).

Preprocessing. Before running keyword extraction on each dataset, we apply standard text preprocessing operations. The documents are first tokenized into words by segtok[9], a Python library for tokenization and sentence splitting. Then, each word is stemmed to reduce it to its base form for comparison purpose by Porter Stemmer from NLTK (Bird et al., 2009), a widely used Python library for text processing. Part-of-speech annotation is carried out using NLTK tagger. To select a candidate phrase set $\mathcal{P}_d$, following the literature (Wan and Xiao, 2008b), we consider contiguous nouns in the document d that form a noun phrase satisfying the regular expression (ADJECTIVE)*(NOUN)+.[10] We then filter the candidates with

[7] All the datasets were fetched from a public data repository for keyword extraction data: https://github.com/LIAAD/KeywordExtractor-Datasets: KPCrowd (Marujo et al., 2013), Inspec (Hulth, 2003), Krapivin2009 (Krapivin et al., 2009), SemEval2017 (Augenstein et al., 2017), kdd (Gollapalli and Caragea, 2014), www (Gollapalli and Caragea, 2014), wiki20 (Medelyan and Witten, 2008), PubMed (Schutz et al., 2008), Schutz2008 (Schutz et al., 2008), citeulike180 (Medelyan et al., 2009), fao30 and fao780 (Medelyan and Witten, 2008), guyen2007 (Nguyen and Kan, 2007), and SemEval2010 (Kim et al., 2010).

[8] We use keyword and keyphrase almost indistinctly, as some datasets contain keyphrases of more than a single token.

[9] https://pypi.org/project/segtok/

[10] While the vast majority of keywords in the considered datasets follow this structure, there are a few cases of different Part-of-Speech tags as keywords, or where this simple formulation can

a stopword list taken from the official YAKE implementation[11] (Campos et al., 2020). Finally, for the statistical methods and the graph-based methods based on them (i.e., LexRank and TFIDFRank), we compute prior statistics including TF , TF-IDF, and LDA by Gensim (Řehůřek and Sojka, 2010) within each dataset.

Baselines. As **statistical** models, we include keyword extraction methods based on TF, TF-IDF, and lexical specificity referred as TF, TF-IDF, and LexSpec[12] respectively.[13] Each model uses its statistics as a score for the individual words and then aggregates them to score the candidate phrases (see § 7.1.1.1). We also add a heuristic baseline which takes the first n phrases as its prediction (FirstN). As **graph-based** models, we compare five distinct methods: TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008a), PositionRank (Florescu and Caragea, 2017), SingleTPR (Sterckx et al., 2015), and TopicRank (Bougouin et al., 2013). Additionally, we propose two extensions of SingleRank, which we call TFIDFRank and LexRank, where a word distribution computed by TF-IDF or lexical specificity is used for $p_b(\cdot)$. As implementations of graph operations such as PageRank and word graph construction, we use NetworkX (Hagberg et al., 2008), a graph analyzer in Python.

	Dataset	FirstN	TF	Lex Spec	TF-IDF	Text Rank	Single Rank	Position Rank	Lex Rank	TF-IDF Rank	Single TPR	Topic Rank
P@5	KPCrowd	35.8	25.3	**39.0**	**39.0**	30.6	30.5	31.8	32.0	32.1	26.9	37.0
	Inspec	31.0	18.9	31.0	31.5	33.2	**33.8**	32.7	32.9	33.3	30.4	31.3
	Krapivin2009	**16.7**	0.1	8.7	7.6	6.6	9.1	14.3	9.7	9.7	7.4	8.5
	Nguyen2007	17.8	0.2	17.2	15.9	13.1	17.3	**20.6**	18.6	18.6	14.0	13.3
	PubMed	9.8	3.6	7.5	6.7	10.1	**10.6**	10.1	8.9	8.8	9.3	7.8
	Schutz2008	16.9	1.6	39.0	38.9	34.0	36.5	18.3	38.9	39.4	14.5	**46.6**
	SemEval2010	15.1	1.5	14.7	12.9	13.4	17.4	**23.2**	16.8	16.6	12.8	16.5
	SemEval2017	30.1	17.0	45.7	**47.2**	41.5	43.0	40.5	46.0	46.4	34.3	36.5
	citeulike180	6.6	9.5	18.0	15.2	23.0	23.9	20.3	23.2	**24.4**	23.7	16.7
	fao30	17.3	16.0	24.0	20.7	26.0	30.0	24.0	29.3	29.3	**32.7**	24.7
	fao780	9.3	3.2	11.7	10.5	12.4	14.3	13.2	13.2	13.1	**14.5**	12.0
	kdd	11.7	7.0	11.2	11.6	10.6	11.5	11.9	11.6	**11.9**	9.4	10.7
	theses100	5.6	0.9	**10.7**	9.4	6.6	7.8	9.3	10.6	9.1	8.3	8.1
	wiki20	13.0	13.0	17.0	21.0	13.0	19.0	14.0	22.0	**23.0**	19.0	16.0
	www	12.2	8.1	11.9	12.2	10.6	11.2	**12.6**	11.6	11.7	10.2	11.2
	AVG	16.6	8.4	20.5	20.0	19.0	21.1	19.8	21.7	**21.8**	17.8	19.8
MRR	KPCrowd	60.1	45.5	**73.6**	72.4	62.4	61.6	64.0	65.8	65.2	50.2	60.7
	Inspec	57.3	33.0	52.4	52.8	51.4	52.4	57.1	53.3	53.7	50.5	**57.8**
	Krapivin2009	**36.1**	1.3	22.9	21.0	18.1	22.2	31.4	23.6	23.8	19.1	21.8
	Nguyen2007	43.0	2.8	38.1	41.2	30.8	34.6	**43.2**	36.4	37.9	29.8	33.7
	PubMed	23.1	13.3	23.5	21.4	**31.7**	30.5	30.6	26.9	26.3	26.0	19.8
	Schutz2008	24.6	8.6	76.6	**76.7**	68.9	70.9	38.5	75.5	76.3	33.7	67.3
	SemEval2010	**49.7**	4.5	35.8	34.6	32.9	35.5	47.8	35.3	36.4	28.7	35.9
	SemEval2017	52.0	32.7	68.6	**68.7**	61.4	63.5	62.4	67.3	67.2	54.3	63.7
	citeulike180	20.9	23.6	55.5	47.7	58.2	62.6	51.0	63.0	**65.7**	62.5	40.3
	fao30	31.1	38.3	61.8	49.1	60.2	70.0	48.6	66.1	67.0	**74.6**	50.6
	fao780	17.0	8.5	39.0	35.9	36.1	38.6	35.9	**39.5**	38.9	38.4	31.6
	kdd	26.1	13.0	27.0	27.8	24.5	26.5	28.1	27.9	**28.8**	18.3	26.2
	theses100	15.1	3.1	**32.5**	31.6	23.2	26.3	24.9	31.6	31.1	26.1	26.9
	wiki20	27.5	27.7	**52.7**	47.7	40.1	45.7	31.1	52.2	46.5	39.6	35.5
	www	29.7	17.1	30.5	**30.6**	26.5	27.6	30.4	29.2	30.1	21.7	27.9
	AVG	34.2	18.2	46.0	44.0	41.8	44.6	41.7	46.2	**46.3**	38.2	40.0

Table 7.2: P@5 and MRR, where the best score in each dataset is highlighted using a bold font.

7.1.2.2 Experimental Results

In this section, we report our main experimental results comparing unsupervised keyword extraction methods. Table 7.2 shows the results obtained by all comparison systems. Results are reported according to standard metrics in keyword extraction and IR: Precision at 5 (P@5) and Mean Reciprocal Rank (MRR). The algorithms in each metric that achieve the best accuracy across datasets are TFIDFRank for P@5, and LexSpec and TF-IDF for MRR. In the averaged metrics over all datasets, lexical specificity and TF-IDF based models (TF-IDF, LexSpec, TFIDFRank, and LexRank) are shown to perform high in general. In particular, the hybrid models LexRank and TFIDFRank achieve the best accuracy on all the metrics, with LexSpec and TF-IDF being competitive in MRR. Overall, despite their simplicity, both lexical specificity and TF-IDF appear to be able to exploit effective features for keyword extraction from a variety of datasets, and perform robustly to domain shifts including document size, format, as well as the source domain. Moreover, TF gives a remarkably low accuracy on every metric and the huge gap between TF and TF-IDF can be interpreted as the improvement given by the normalization provided by the inverse document frequency. However, the inverse document frequency normalization relies on a corpus partition, which may not be available in all cases. On the other hand, a measure such as lexical specificity only needs overall term frequencies, which may have advantages in practical settings. In the following sections we perform a more in-depth analysis on these results and the global perfor-

miss a correct candidate. Nonetheless, our experimental setting is focused on comparing keyword extraction measures, within the same preprocessing framework.

[11] https://github.com/LIAAD/yake

[12] For lexical specificity, we follow the implementation of Camacho-Collados et al. (2016).

[13] As mentioned in § 7.1.1.1, we do not include YAKE (Campos et al., 2020) as our experiments are focused on analyzing single features on their own in a unified setting. YAKE utilizes a unified preprocessing and a combination of various textual features, which are out of scope in this paper.

Prior	Model	Time prior	Time total	Time per doc
TF	TF	10.2	11.5	0.0058
	LexSpec		12.1	0.0061
	LexRank		25.5	0.0128
TF-IDF	TF-IDF	10.3	22.4	0.0112
	TFIDFRank		26.5	0.0133
LDA	SingleTPR	16.2	29.4	0.0147
-	FirstN	-	11.5	0.0058
	TextRank		14.9	0.0075
	SingleRank		15.0	0.0075
	PositionRank		15.0	0.0075
	TopicRank		19.0	0.0095

Table 7.3: Average clock time (sec.) to process the Inspec dataset over 100 independent trials.

mance of each type model.

Execution time. In terms of efficiency for each algorithm, we report the average process time over 100 independent trials on the Inspec dataset in Table 7.3, which also includes the time to compute each statistical prior over the dataset. In general, none of the models perform very slowly. Not surprisingly, statistical models are faster than graph-based models due to the overhead introduced by the PageRank algorithm, although as a drawback they need to perform prior statistical computations for each dataset beforehand.

7.2 T-NER

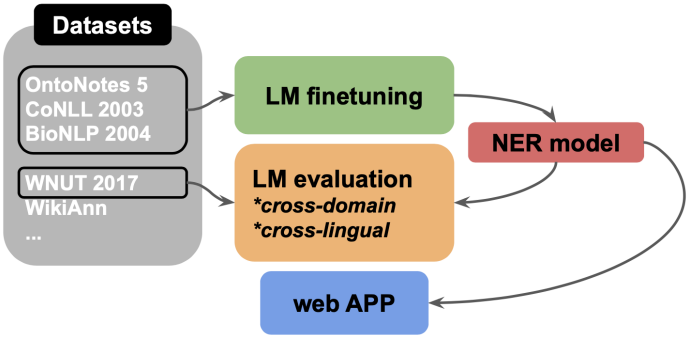

Figure 7.2: System overview of T-NER.

In this paper, we introduce **T-NER**, an open-source Python library for cross-domain analysis for NER with pre-trained Transformer-based LMs. Figure 7.2 shows a brief overview of our library and its functionalities. The library facilitates NER experimental design including easy-to-use features such as model training and evaluation. Most notably, it enables to organize cross-domain analyses such as training a NER model and testing it on a different domain, with a small configuration. We also report initial experiment results, by which we show that although cross-domain NER is challenging, if it has an access to new domains, LM can successfully learn new domain knowledge. The results give us an insight that LM is capable to learn a variety of domain knowledge, but an ordinary fine-tuning scheme on single dataset most likely causes overfitting and results in poor domain generalization.

As a system design, T-NER is implemented in Pytorch (Paszke et al., 2019) on top of the Transformers library. Moreover, the interfaces of our training and evaluation modules are highly inspired by Scikit-learn (Pedregosa et al., 2011b), enabling an interoperability with recent models as well as integrating them in an intuitive way. In addition to the versatility of our toolkit for NER experimentation, we also in-

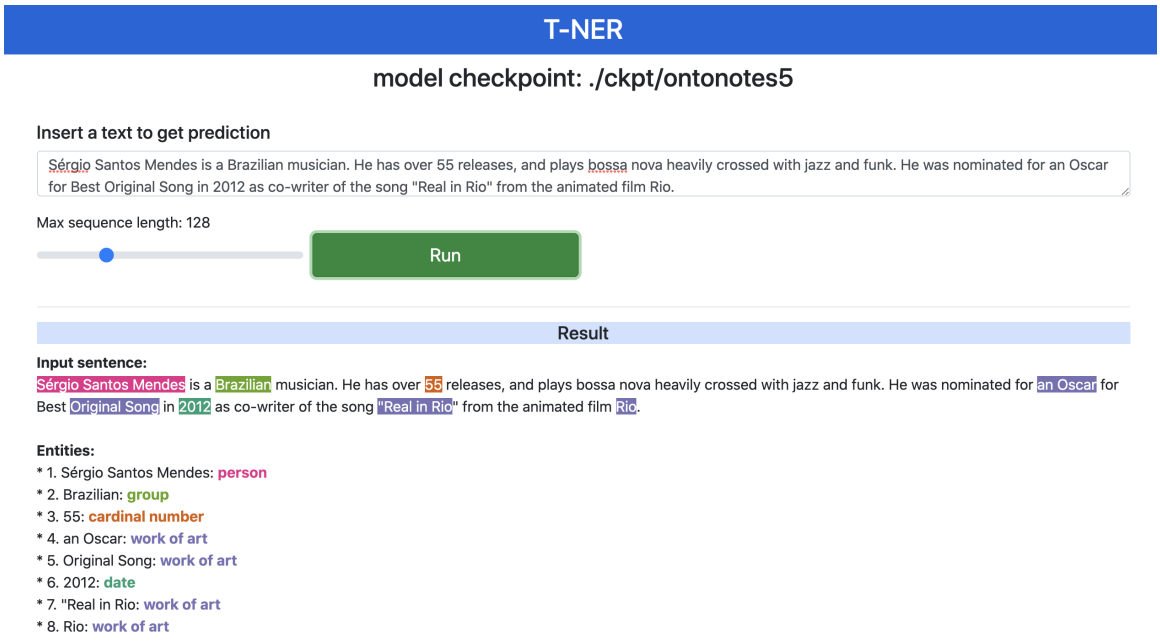

Figure 7.3: A screenshot from the demo web app. In this example, the NER transformer model is fine-tuned on OntoNotes 5 and a sample sentence is fetched from Wikipedia (en.wikipedia.org/wiki/Sergio_Mendes).

clude an online demo and robust pre-trained models trained across domains. In the following sections, we provide a brief overview about NER in § 7.2.1, explain the system architecture of T-NER with a few basic usages in § 7.2.2 and describe experiment results on cross-domain transfer with our library in § 7.2.3.

7.2.1 Named Entity Recognition

Given an arbitrary text, the task of NER consists of detecting named entities and identifying their type. For example, given a sentence *"Dante was born in Florence."*, a NER model are would identify *"Dante"* as a person and *"Florence"* as a location. Traditionally, NER systems have relied on a classification model on top of hand-

engineered feature sets extracted from corpora (Ratinov and Roth, 2009; Collobert et al., 2011), which was improved by carefully designed neural network approaches (Lample et al., 2016; Chiu and Nichols, 2016; Ma and Hovy, 2016). This paradigm shift was mainly due to its efficient access to contextual information and flexibility, as human-crafted feature sets were no longer required. Later, contextual representations produced by pre-trained LMs have improved the generalization abilities of neural network architectures in many NLP tasks, including NER (Peters et al., 2018b; Devlin et al., 2019).

In particular, LMs see millions of plain texts during pre-training, a knowledge that then can be leveraged in downstream NLP applications. This property has been studied in the recently literature by probing their generalization capacity (Hendrycks et al., 2020; Aharoni and Goldberg, 2020; Desai and Durrett, 2020; Gururangan et al., 2020). When it comes to LM generalization studies in NER, the literature is more limited and mainly restricted to in-domain (Agarwal et al., 2021) or multilingual settings (Pfeiffer et al., 2020a; Hu et al., 2020b). Our library facilitates future research in cross-domain and cross-lingual generalization by providing a unified benchmark for several languages and domain as well as a straightforward implementation of NER LM fine-tuning.

7.2.2 T-NER: An Overview

A key design goal was to create a self-contained universal system to train, evaluate, and utilize NER models in an easy way, not only for research purpose but also practical use cases in industry. Moreover, we provide a demo web app (Figure 7.3) where users can get predictions from a trained model given a sentence interactively.

This way, users (even those without programming experience) can conduct qualitative analyses on their own or existing pre-trained LMs.

In the following we provide details on the technicalities of the package provided, including details on how to train and evaluate any LM-based architecture. Our package, T-NER, allows practitioners in NLP to get started working on NER with a few lines of code while diving into the recent progress in LM fine-tuning. We employ Python as our core implementation, as is one of the most prevailing languages in the machine learning and NLP communities. Our library enables Python users to access its various kinds of features such as model training, in- and cross-domain model evaluation, and an interface to get predictions from trained models with minimum effort.

7.2.2.1 Datasets

For model training and evaluation, we compiled nine public NER datasets from different domains, unifying them into same format: OntoNotes5 (Hovy et al., 2006), CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003), WNUT 2017 (Derczynski et al., 2017), WikiAnn (Pan et al., 2017), FIN (Salinas Alvarado et al., 2015), BioNLP 2004 (Collier et al., 2004), BioCreative V CDR[14] (Wei et al., 2015), MIT movie review semantic corpus,[15] and MIT restaurant review.[16] These unified datasets are also made available as part of our T-NER library. Except for WikiAnn that

[14]The original dataset consists of long documents which cannot be fed on LM because of the length, so we split them into sentences to reduce their size.

[15]The movie corpus includes two datasets (*eng* and *trivia10k13*) coming from different data sources. While both have been integrated into our library, we only used the largest *trivia10k13* in our experiments.

[16]The original MIT NER corpora can be downloaded from `https://groups.csail.mit.edu/sls/downloads/`.

Name	Domain	Entity types	Data size
OntoNotes5	News, Blog, Dialogue	18	59,924/8,582/8,262
CoNLL 2003	News	4	14,041/3,250/3,453
WNUT 2017	SNS	6	1,000/1,008/1,287
WikiAnn	Wikipedia (282 languages)	3	20,000/10,000/10,000
FIN	Finance	4	1,164/-/303
BioNLP 2004	Biochemical	5	18,546/-/3,856
BioCreative V	Biomedical	5	5,228/5,330/5,865
MIT Restaurant	Restaurant review	8	7,660/-/1,521
MIT Movie	Movie review	12	7,816/-/1,953

Table 7.4: Overview of the NER datasets used in our evaluation and included in T-NER. Data size is the number of sentence in training/validation/test set.

contains 282 languages, all the datasets are in English, and only the MIT corpora are lowercased. As MIT corpora are commonly used for slot filling task in spoken language understanding (Liu and Lane, 2017), the characteristics of the entities and annotation guidelines are quite different from the other datasets, but we included them for completeness and to analyze the differences across datasets.

Table 7.4 shows statistics of each dataset. In § 7.2.3, we train models on each dataset, and assess the in- and cross-domain accuracy over them.

Dataset Format and Customization. Users can utilize their own datasets for both model training and evaluation by formatting them into the IOB scheme (Tjong Kim Sang and De Meulder, 2003) which we used to unify all datasets. In the IOB format, all data files contain one word per line with empty lines representing sentence boundaries. At the end of each line there is a tag which states whether the current word is inside a named entity or not. The tag also encodes the type of named entity. Here is an example from CoNLL 2003:

```
EU B-ORG
rejects O
German B-MISC
call O
to O
boycott O
British B-MISC
lamb O
. O
```

7.2.2.2 Model Training

We provide modules to facilitate LM fine-tuning on any given NER dataset. Following Devlin et al. (2019), we add a linear layer on top of the last embedding layer in each token, and train all weights with cross-entropy loss. The model training component relies on the Huggingface transformers library, one of the largest Python frameworks for distributing pre-trained LM checkpoint files. Our library is therefore fully compatible with the Transformers framework: once new model was deployed on the Transformer hub, one can immediately try those models out with our library as a NER model.

The instance of model training in a given dataset[17] can be used in an intuitive way as displayed below:

```
from tner import TrainTransformersNER
model = TrainTransformersNER(
```

[17]To use custom datasets, the path to a custom dataset folder can simply be included in the dataset argument.

```
            dataset="ontonotes5",
            transformer="RoBERTa\textsubscript{BASE}")
    model.train()
```

With this sample code, we would fine-tune RoBERTa$_{\text{BASE}}$ on the OntoNotes5 dataset. We also provide an easy extension to train on multiple datasets at the same time:

```
    TrainTransformersNER(
        dataset=[
            "ontonotes5", "wnut2017"
            ],
        transformer="RoBERTa\textsubscript{BASE}")
```

Once training is completed, checkpoint files with model weights and other statistics are generated. These are automatically organized for each configuration and can be easily uploaded to the Hugging Face model hub. Ready-to-use code samples can be found in our Google Colab notebook[18], and details for additional options and arguments are included in the GitHub repository.

7.2.2.3 Model Evaluation

Once a NER model is trained, users may want to test the models in the same dataset or a different one to assess its general performance across domains. To this end, we implemented flexible evaluation modules to facilitate cross-domain evaluation comparison, which is also aided by the unification of datasets into the same format (see

[18] https://colab.research.google.com/drive/1AlcTbEsp8W11yflT7SyT0L4C4HG6MXYr?usp=sharing

§ 7.2.2.1) with a unique label reference lookup. The basic usage of the evaluation module is described below.

```
from tner import TrainTransformersNER
model = TrainTransformersNER(
    "path-to-model-checkpoint"
    )
model.test("ontonotes5")
```

Here, the model would be tested on OntoNotes5 dataset, and it could be evaluated on any other test set including custom dataset.

7.2.3 Evaluation

In this section, we assess the reliability of T-NER with experiments in standard NER datasets.

7.2.3.1 Experimental Setting

Implementation Details. Through the experiments, we use XLM-R (Conneau et al., 2019), which has shown to be one of the most reliable multi-lingual pre-trained LMs for discriminative tasks at the moment. In all experiments we make use of the default configuration and hyperpameters of Huggingface's XLM-R implementation. For WikiAnn/ja (Japanese), we convert the original character-level tokenization into proper morphological chunk by MeCab[19].

[19] https://pypi.org/project/mecab-python3/

Evaluation metrics and protocols As customary in the NER literature, we report *span micro-F1 score* computed by seqeval[20], a Python library to compute metrics for sequence prediction evaluation. We refer to this F1 score as *type-aware* F1 score to distinguish it from the the type-ignored metric used to assess the cross-domain performance, which we explain below.

In a cross-domain evaluation setting, the *type-aware* F1 score easily fails to represent the cross-domain performance if the granularity of entity types differ across datasets. For instance, the MIT restaurant corpus has entities such as *amenity* and *rating*, while *plot* and *actor* are entities from the MIT movie corpus. Thus, we report *type-ignored* F1 score for cross-domain analysis. In this *type-ignored* evaluation, the entity type from both of predictions and true labels is disregarded, reducing the task into a simpler entity span detection task. This evaluation protocol can be customized by the user at test time.

7.2.3.2 Experimental Results

We conduct three experiments on the nine datasets described in Table 7.4: (i) in-domain evaluation, (ii) cross-domain evaluation, and (iii) cross-lingual evaluation. While the first experiment tests our implementation in standard datasets, the second experiment is aimed at investigating the cross-domain performance of transformer-based NER models. Finally, as a direct extension of our evaluation module, we show the zero-shot cross-lingual performance of NER models on the WikiAnn dataset.

[20] https://pypi.org/project/seqeval/

Dataset	XLM-R_{BASE}	XLM-R_{LARGE}	SoTA
OntoNotes5	89.0	89.1	92.1
CoNLL 2003	90.8	92.9	94.3
WNUT 2017	52.8	58.5	50.3
FIN	81.3	76.4	82.7
BioNLP 2004	73.4	74.3	77.4
BioCreative V	88.0	88.6	89.9
MIT Restaurant	79.4	79.6	-
MIT Movie	69.9	71.2	-
WikiAnn/en	82.7	84.0	84.8
WikiAnn/ja	83.8	86.5	73.3
WikiAnn/ru	88.6	90.0	91.4
WikiAnn/es	90.9	92.1	-
WikiAnn/ko	87.5	89.6	-
WikiAnn/ar	88.9	90.3	-

Table 7.5: In-domain *type-aware* F1 score for test set on each dataset with current SoTA. SoTA on each dataset is attained from the result of *BERT-MRC-DSC* (Li et al., 2019b) for OntoNotes5, *LUKE* (Yamada et al., 2020) for CoNLL 2003, *Cross-Weigh* (Wang et al., 2019c) for WNUT 2017, (Pfeiffer et al., 2020a) for WikiAnn (en, ja, ru, es, ko, ar), (Salinas Alvarado et al., 2015) for FIN, (Lee et al., 2020) for BioNLP 2004, (Nooralahzadeh et al., 2019) for BioCreative V and (Pfeiffer et al., 2020a) for WikiAnn/en.

In-domain Results. The main results are displayed in Table 7.5, where we report the *type-aware* F1 score from XLM-R_{BASE} and XLM-R_{LARGE} models along with current SoTA. One can confirm that our framework with XLM-R_{LARGE} achieves a comparable SoTA score, even surpassing it in the WNUT 2017 dataset. In general, XLM-R_{LARGE} performs consistently better than XLM-R_{BASE} but, interestingly, the base model performs better than large on the FIN dataset. This can be attributed to the limited training data in this dataset, which may have caused overfitting in the large model.

Train\Test	ontonotes	conll	wnut	wiki	bionlp	bc5cdr	fin	avg
ontonotes	**91.6**	65.4	53.6	47.5	0.0	0.0	18.3	40.8
conll	62.2	**96.0**	69.1	61.7	0.0	0.0	22.7	35.1
wnut	41.8	85.7	**68.3**	54.5	0.0	0.0	20.0	31.7
wiki	32.8	73.3	53.6	**93.4**	0.0	0.0	12.2	29.6
bionlp	0.0	0.0	0.0	0.0	**79.0**	0.0	0.0	8.7
bc5cdr	0.0	0.0	0.0	0.0	0.0	**88.8**	0.0	9.8
fin	48.2	73.2	60.9	58.9	0.0	0.0	**82.0**	38.1
all	90.9	93.8	60.9	91.3	78.3	84.6	75.5	81.7

Table 7.6: *Type-ignored* F1 score in cross-domain setting over non-lower-cased English datasets. We compute average of accuracy in each test set, named as avg. The model trained on all datasets listed here, is shown as all.

Generally, it can be expected to get better accuracy with domain-specific or larger LMs that can be integrated into our library. Nonetheless, our goal for these experiments were not to achieve SoTA but rather to provide a competitive and easy-to-use framework.

Cross-domain Results. In this section, we show cross-domain evaluation results on the English datasets[21]: OntoNotes5 (ontonotes), CoNLL 2003 (conll), WNUT 2017 (wnut), WikiAnn/en (wiki), BioNLP 2004 (bionlp), and BioCreative V (bc5cdr), FIN (fin). We also report the accuracy of the same XLM-R model trained over a combined dataset resulting from concatenation of all the above datasets.

In Table 7.6, we present the *type-ignored* F1 results across datasets. Overall cross-domain scores are not as competitive as in-domain results. This gap reveals the difficulty of transferring NER models into different domains, which may also be attributed to different annotation guidelines or data construction procedures across

[21]We excluded the MIT datasets in this setting since they are all lowercased.

Train\Test	en	ja	ru	ko	es	ar
en	**84.0**	46.3	73.1	58.1	71.4	53.2
ja	53.0	**86.5**	45.7	57.1	74.5	55.4
ru	60.4	53.3	**90.0**	68.1	76.8	54.9
ko	57.8	62.0	68.6	**89.6**	66.2	57.2
es	70.5	50.6	75.8	61.8	**92.1**	62.1
ar	60.1	55.7	55.7	70.7	79.7	**90.3**

Table 7.7: Cross-lingual *type-aware* F1 results on various languages for the WikiAnn dataset.

datasets. Especially, training on the bionlp and bc5cdr datasets lead to a null accuracy when they are evaluated on other datasets, as well as others evaluated on them. Those datasets are very domain specific dataset, as they have entities such as *DNA*, *Protein*, *Chemical*, and *Disease*, which results in a poor adaptation to other domains. On the other hand, there are datasets that are more easily transferable, such as wnut and conll. The wnut-trained model achieves 85.7 on the conll dataset and, surprisingly, the conll-trained model actually works better than the wnut-trained model when evaluated on the wnut test set. This could be also attributed to the data size, as wnut only has 1,000 sentences, while conll has 14,041. Nevertheless, the fact that ontonotes has 59,924 sentences but does not perform better than conll on wnut reveals a certain domain similarity between conll and wnut.

Finally, the model trained on the training sets of all datasets achieves a *type-ignored* F1 score close to the in-domain baselines. This indicates that a LM is capable of learning representations of different domains. Moreover, leveraging domain similarity as explained above can lead to better results as, for example, distant datasets such as bionlp and bc5cdr surely cause performance drops. This is an example of the type of experiments that could be facilitated by T-NER, which we leave for future work.

Cross-lingual Results. Finally, we present some results for zero-shot cross-lingual NER over the WikiAnn dataset, where we include six distinct languages: English (en), Japanese (ja), Russian (ru), Korean (ko), Spanish (es), and Arabic (ar). In Table 7.7, we show the cross-lingual evaluation results. The diagonal includes the results of the model trained on the training data of the same target language. There are a few interesting findings. First, we observe a high correlation between Russian and Spanish, which are generally considered to be distant languages and do not share the alphabet. Second, Arabic also transfers well to Spanish which, despite the Arabic (lexical) influence on the Spanish language (Stewart et al., 1999), are still languages from distant families.

Clearly, this is a shallow cross-lingual analysis, but it highlights the possibilities of our library for research in cross-lingual NER. Recently, Hu et al. (2020a) proposed a compilation of multilingual benchmark tasks including the WikiAnn datasets as a part of it, and XLM-R proved to be a strong baseline on multilingual NER. This is in line with the results of Conneau et al. (2020b), which showed a high capacity of zero-shot cross-lingual transferability. On this respect, Pfeiffer et al. (2020b) proposed a language/task specific adapter module that can further improve cross-lingual adaptation in NER. Given the possibilities and recent advances in cross-lingual LMs in recent years, we expect our library to help practitioners to experiment and test these advances in NER.

7.3 TweetNLP

Today's society cannot be understood without the role of social media. Online users connect more and more via platforms that enable content sharing, either generic or around specific topics, and do this by means of text-only messages, or augmenting them with multimedia content such as pictures, audio or video. As such, these platforms have been used to understand user, group and organization-wide behaviours (Yang et al., 2021; Hu et al., 2021). In particular, Twitter, which is the main platform studied in this paper, has long been an important resource for understanding society at large (Weller et al., 2013). Twitter is interesting for NLP because it embodies many features that are natural in spontaneous and ever-evolving fast-paced communication. However, the majority of NLP research focuses on optimizing model development against training data and evaluation benchmarks which are, at worst, reasonably clean (e.g., news articles, blog posts or Wikipedia). Consequently, when deployed *in the wild*, features such as noisiness, multilinguality, immediacy, slang, technical jargon, lack of context, platform-specific restrictions on message length, emoji and other modalities, etc. become core communicative variables that need to be factored in. Indeed, even traditional NLP tasks such as normalization (Han and Baldwin, 2011; Baldwin et al., 2015), part-of-speecg tagging (Derczynski et al., 2013), sentiment analysis (Poria et al., 2020) or NER (Ritter et al., 2011; Baldwin et al., 2013) have been shown to produce suboptimal results in the context of social media.

Given the above, we put forward TweetNLP (`tweetnlp.org`), which offers a full-fledged NLP platform specialized in Twitter. The backbone of TweetNLP consists of LMs that have been trained on Twitter (Barbieri et al., 2020, 2022; Loureiro

et al., 2022). Then, these specialized LMs have been further fine-tuned for specific NLP tasks on Twitter data. These models have already proved highly popular, with thousands of downloads from the Hugging Face model hub every month (Wolf et al., 2020). Most notably, the sentiment analysis model has been the most downloaded model in the Hugging Face model hub in January 2022, with over 15M downloads. Similarly, the TweetEval benchmark, in which most task-specific Twitter models are fine-tuned, has been the second most downloaded dataset in April 2022, with over 150K downloads. TweetNLP integrates all these resources into a single platform. With a simple Python API, TweetNLP offers an easy-to-use way to leverage cutting-edge NLP models in a variety of social media tasks. Despite the trend of ever-larger LMs (Brown et al., 2020), TweetNLP is more focused on the general user and applicability, and therefore integrates base models which are easily run in standard computers or on free cloud services. Finally, all models can be accessed from an interactive online demo, offering anyone the possibility to test models and perform real-time analysis on Twitter.

7.3.1 Models and Functionalities

TweetNLP is versatile in that it covers a wide range of tasks and applications. The backbone of TweetNLP are transformer-based LMs, which are covered in § 7.3.1.1. The concrete NLP tasks integrated in TweetNLP are presented in § 7.3.1.2. Finally, in § 7.3.1.3 we present embeddings used to represent words and tweets.

7.3.1.1 LMs

LMs are at the core of TweetNLP. Instead of relying on general-purpose models (Devlin et al., 2019) or training a LM from scratch (Nguyen et al., 2020), we start from RoBERTa and XLM-R (Conneau et al., 2019) checkpoints and continue pre-training on Twitter-specific corpora. This was shown to be generally more reliable for the amount of text analysed in Barbieri et al. (2020). Given our aim for democratizing the usage of specialized LMs for social media, another important feature of TweetNLP is the relatively small size of the LMs. To this end, all LMs rely on the equivalent of a $\text{RoBERTa}_{\text{BASE}}$ or $\text{XLM-R}_{\text{BASE}}$ architecture. These models are efficient on standard hardware and free-tiers of cloud computing services, with reasonable speed even without GPU support.

TweetEval (Barbieri et al., 2020). This model was initially released as part of the TweetEval project. It is based on a $\text{RoBERTa}_{\text{BASE}}$ architecture using the original model as an initial check point (Liu et al., 2019). Later, this LM was further pre-trained on a corpus of 60M tweets from May 2018 to August 2019.

TimeLMs (Loureiro et al., 2022). This model is initially trained on the same Twitter corpus used by Barbieri et al. (2020). The main difference lies on a few preprocessing improvements applied to the underlying corpus, including measures to reduce potential spam and near duplicates, and more recent corpora used for continual pre-training. The overall quantity of tweets is therefore larger, as the model is regularly updated (every 3 months) with a fixed number of additional tweets. The most recently released TimeLMs model to date is pre-trained on 132M tweets until the end of June 2022.

XLM-T (Barbieri et al., 2022). This model was trained on 198M tweets on over thirty languages from May 2018 to March 2020, following a similar strategy to Barbieri et al. (2020). In this case, the initial checkpoint was XLM-R_{BASE} (Conneau et al., 2020a).

7.3.1.2 Supported tasks

In the following we describe the tasks supported by TweetNLP. For the tweet classification tasks included in TweetEval, and for topic classification, we simply fine-tune the models described above on the corresponding datasets, as described in Barbieri et al. (2020).

Sentiment Analysis. The sentiment analysis task integrated in TweetNLP consists of predicting the sentiment of a tweet with one of the three following labels: positive, neutral or negative. The base dataset for English is the unified TweetEval version of the Semeval-2017 dataset from the task on *Sentiment Analysis in Twitter* (Rosenthal et al., 2017). Moreover, for the languages other than English we include the datasets integrated in UMSAB (Barbieri et al., 2022), namely Arabic (Rosenthal et al., 2017), French (Benamara et al., 2017), German (Cieliebak et al., 2017), Hindi (Patra et al., 2015), Italian (Barbieri et al., 2016), Portuguese (Brum and Volpe Nunes, 2018), and Spanish (Díaz-Galiano et al., 2018).

Emotion Recognition. Given a tweet, this task consists of associating it with its most appropriate emotion. As a reference dataset we use the SemEval 2018 task on *Affect in Tweets* (Mohammad et al., 2018), simplified to only the four emotions

used in TweetEval: anger, joy, sadness and optimism.

Emoji Prediction. The goal of emoji prediction is to predict the final emoji on a given tweet. The dataset used to fine-tune our models is the TweetEval adaptation from the SemEval 2018 task on *Emoji Prediction* (Barbieri et al., 2018), including 20 emoji as labels.

Irony Detection. This is a binary classification task that aims at detecting whether a tweet is ironic or not. It is based on the *Irony Detection* dataset from the SemEval 2018 task (Van Hee et al., 2018).

Hate Speech Detection. The hate speech dataset consists of detecting whether a tweet is hateful towards women or immigrants. It is based on the *Detection of Hate Speech* task at SemEval 2019 (Basile et al., 2019).

Offensive Language Identification. The task consist of identifying any form of offensive language in a tweet. The dataset is based on the SemEval 2019 task on *Identifying and Categorizing Offensive Language in Social Media* (Zampieri et al., 2019).

Stance Detection. Given a target topic and a tweet, stance detection consists of assessing whether the author of the tweet has a positive, neutral or negative position towards the target. The dataset considered was initially released for the SemEval 2016 task on *Detecting Stance in Tweets* (Mohammad et al., 2016).

Topic Classification. The aim of this task is, given a tweet, assign topics related to its content. The task is formulated as a supervised multi-label classification problem where each tweet is assigned one or more topics from a total of 19 available topics. The topics were carefully curated based on Twitter trends with the aim to be broad and general, consisting of classes such as: *arts and culture*, *music*, or *sports*. The underlying tweet topic classification dataset contains over 10K manually-labeled tweets (Antypas et al., 2022).

Named Entity Recognition. The goal of NER is to find entities and identify their entity types in a given sentence. The underlying Twitter NER dataset is composed of over 10K tweets which were annotated (internally) with seven entity types (Ushio et al., 2022b).

7.3.1.3 Embeddings

In addition to the LMs and their supported tasks, we also release high-quality vector representation models for words and tweets, i.e., *embeddings* (Pilehvar and Camacho-Collados, 2020). These relatively low-dimensional vector representations can be exploited for a different range of applications and analyses such as word/tweet similarity or tweet retrieval, to name a few.

Word Embeddings. TweetNLP word embeddings are based on fastText and trained on the same corpora used to train the LMs described in § 7.3.1.1. In particular, we trained two sets of embeddings: (1) a monolingual English model trained with the TimeLMs Twitter corpus until the end of 2021; and (2) a multilingual

model trained with the Twitter corpus used for XLM-T. Both models were trained using the official fastText package with 300 dimensions, minimum n-gram size 2, maximum n-gram size 12, and remaining parameters set to defaults.

Tweet Embeddings. For tweet embeddings, we pulled tweet-reply pairs from the Twitter API and trained contrastive embeddings with an InfoNCE loss (Oord et al., 2018). For tweets with multiple replies, we randomly sampled one reply. In training, one mini-batch is composed of a list of tweet-reply pairs. The tweet-reply pairs are regarded as positive samples; the enumeration of all other possible combination of tweet-reply, tweet-tweet, and tweet-reply pairs are regarded as negative samples. The contrastive InfoNCE loss then pulls positive pair representations close while pushes negative representations away from each other. Training was performed on 1.1M tweet-reply pairs, and we collected a separate tweet-reply set of 10k pairs for selecting the model checkpoint.

7.3.2 TweetNLP Python library

The TweetNLP Python library has been integrated into pypi[22] and therefore is easily accessible and can be installed from pip ("pip install tweetnlp"). All the details on how to use TweetNLP are in the associated Github repository, which is released fully open-source: https://github.com/cardiffnlp/tweetnlp. Once installed, loading and using a fine-tuned model on any specific task can be done as follows.

```
from tweetnlp import load
tweet = "I love Paris!!"
```

[22] https://pypi.org/project/tweetnlp/

```
# Sentiment Analysis
model = load('sentiment')
model.sentiment(tweet)
# Tweet Embeddings
model = load('sentence_embedding')
model.embedding(tweet)
# Masked Language Model
model = load('language_model')
tweet = "I love <mask>!!"
model.mask_prediction(tweet)
```

With the *load* statement, the associated fine-tuned LMs are loaded in the background. Users can then get the predictions for any given sentence or tweet with a simple pre-defined function (e.g., *.sentiment* or *.predict*). Custom loading of existing fine-tuned LMs not included in TweetNLP is also possible. The same functionalities apply to all the other tasks described in § 7.3.1.2.

7.3.2.1 Tutorials

In addition to the Python library presented in the previous section, TweetNLP offers access to the underlying Python code structured in instructive Google Colab notebooks with starter code and examples (https://tweetnlp.org/get-started/). These notebooks are aimed at users with varying degrees of experience in NLP and social media processing. In the following we list the currently existing tutorials and a brief description:

Introduction to TweetNLP. In this initial introduction, users learn how to use the TweetNLP Python library to make use of specialized models in social media for a wide variety of tasks from sentiment analysis to named entity recognition.

Getting Data from Twitter. This notebook helps users understand the Twitter API[23] and how to interact with it. More importantly, there are concrete examples on how to retrieve data (i.e. tweets) from Twitter, usually given a hashtag or a keyword.

Custom Fine-tuning. In this notebook users can learn to fine-tune any given LM on a specific task (e.g. sentiment analysis). For this, we will take advantage of the TweetEval task data and unified format (Barbieri et al., 2020). Additionally, users can learn how to easily evaluate LMs on TweetEval.

Word Embeddings. With this notebook users can learn how to train their own word embeddings on custom data using Gensim[24] (Řehůřek and Sojka, 2010). The notebook also includes examples on how to get similarity scores from Twitter-specific word embeddings, or how to obtain the nearest neighbour words from a given input word.

LMs Over Time. This notebook leverages the TimeLMs library (Loureiro et al., 2022). Users can learn how to make use of LMs that have been trained in short periods of time since 2019 until recently.

[23] https://developer.twitter.com/en/docs/twitter-api
[24] https://radimrehurek.com/gensim/

Tweet Embeddings. This notebook contains examples on how to transform a tweet into a vector (embedding) and how these enable important applications such as tweet similarity and retrieval.

7.3.3 Evaluation

In this section, we provide experimental results of the default models integrated into TweetNLP.

7.3.3.1 Experimental setting

Datasets. For the evaluation we utilized all the train/validation/test splits described in § 7.3.1.2. In particular, we relied on the TweetEval-released datasets for all tweet classification tasks except for topic classification.

Default TweetNLP LMs. While in TweetNLP all Twitter-specific LMs are included, we use as a default (1) TimeLMs trained until December 2021 for English and (2) XLM-T for the languages other than English and multilingual tasks. These models are then fine-tuned to the corresponding tasks as described in § 7.3.1.2.

Comparison Systems. We report the performance of all original TweetEval baselines (Barbieri et al., 2020): a frequency-based SVM classifier, fastText, a bidirectional LSTM, RoBERTa$_{\text{BASE}}$, a RoBERTa$_{\text{BASE}}$ model trained on Twitter from scratch (RoBERTa$_{\text{Twitter}}$) and the original TweetEval RoBERTa$_{\text{BASE}}$ model. As

	Emoji	Emotion	Hate	Irony	Offensive	Sentiment	Stance	Topic	NER
SVM	29.3	64.7	36.7	61.7	52.3	62.9	67.3	30.5	-
fastText	25.8	65.2	50.6	63.1	73.4	62.9	65.4	24.0	-
BiLSTM	24.7	66.0	52.6	62.8	71.7	58.3	59.4	27.0	-
$\text{RoBERTa}_{\text{BASE}}$	30.9	76.1	46.6	59.7	79.5	71.3	68.0	50.1	58.0
$\text{RoBERTa}_{\text{Twitter}}$	29.3	72.0	46.9	**65.4**	77.1	69.1	66.7	-	-
TweetEval	31.4	78.5	52.3	61.7	80.5	72.6	69.3	56.8	56.8
BERTweet	33.4	79.3	**56.4**	*82.1**	79.5	73.4	71.2	52.7	58.7
TimeLMs-21	**34.0**	**80.2**	55.1	64.5	**82.2**	**73.7**	**72.9**	**58.8**	**59.7**
Metric	M-F1	M-F1	M-F1	$F^{(i)}$	M-F1	M-Rec	AVG (F)	M-F1	M-F1

Table 7.8: Test results in the nine TweetNLP-supported tasks.

another baseline we include BERTweet (Nguyen et al., 2020), trained on almost 1 billion tweets from 2013 to 2019.

Language Model Fine-tuning. Fine-tuning is performed on the training sets of each corresponding dataset, using their corresponding development sets for validation. We followed TweetEval training protocols for tweet classification, where only the learning rate and number of epochs are tuned (Barbieri et al., 2020). All reported results for LMs are based on an average of three runs.

7.3.3.2 Experimental Results

Table 7.8 shows the main results of our TweetNLP default LM and comparison systems on nine Twitter-based tasks.[25] The default TimeLMs-21 model achieves the overall results on most tasks, especially comparing it with a comparable general-purpose $\text{RoBERTa}_{\text{BASE}}$ model. In the following we also provide details of our experimental results on languages other than English, and for the integrated word and tweet embedding models.

[25]The BERTweet result on Irony is marked with * as their pre-training corpus overlapped with the Irony dataset, which was constructed using distant supervision.

	Arabic	English	French	German	Hindi	Italian	Portuguese	Spanish	ALL
fastText	45.9	50.8	54.8	59.5	37.0	54.6	55.0	50.0	51.0
XLM-R	64.3	68.5	70.5	72.8	53.3	68.6	69.7	66.0	66.7
XLM-T	**66.8**	**70.6**	**71.1**	**77.3**	**56.3**	**69.0**	**75.4**	**68.5**	**67.9**

Table 7.9: Sentiment analysis results (Macro-F1) on the UMSAB unified benchmark. XLM-R and TweetNLP models (XLM-T) are fine-tuned on the training sets of all languages.

Multilingual Sentiment Analysis Results. In addition to the English evaluation, we report results on multilingual sentiment analysis (Table 7.9). The evaluation is performed on the UMSAB multingual sentiment analysis benchmark (Barbieri et al., 2022). For this evaluation we compare XLM-T fine-tuned on all the language-specific training sets of UMSAB with XLM-R (Conneau et al., 2020a) using the same fine-tuning strategy. As an additional indicative baseline, we include fastText trained on the language-specific training sets. As can be observed, our domain-specific XLM-T LM achieves the best overall results in all languages, further reinforcing the importance of in-domain LM training.

Word Embedding Results. As a sanity check to verify the quality of the word embeddings, we simply test them on standard word similarity datasets: The WS-Sim similarity and WS-Rel relatedness subsets (Agirre et al., 2009) from WordSim-353 (Finkelstein et al., 2002), SemEval-2017 (Camacho-Collados et al., 2017) and MEN (Bruni et al., 2014). Then, we compared the results with the pre-trained fastText model trained on the Common Crawl, and Wikipedia. According to Spearman's correlation, the results of our Twitter embeddings were 0.77 (WS-Sim), 0.72 (WS-Rel), 0.69 (SemEval), and 0.79 (MEN).[26] In contrast, the pre-trained fastText

[26]While not directly comparable given the different sizes, we also compared with our previously-released Twitter-specific 100-dimensional fastText embeddings (Camacho-Collados et al., 2020). The results for these embeddings were consistently lower: 0.65 (WS-Sim), 0.43 (WS-Rel), 0.52 (SemEval), and 0.76 (MEN).

Model	Retrieval	STS
Sentence-BERT	6.1	77.0
`all-mpnet-base-v2`	15.8	**83.4**
Mirror-RoBERTa	8.8	79.6
SimCSE-RoBERTa	9.2	80.3
TweetNLP (Tweet-embeddings)	**26.7**	70.7

Table 7.10: Results of sentence and tweet embedding models on tweet-reply retrieval and the STS-benchmark.

Common Crawl results were 0.84 (WS-Sim), 0.64 (WS-Rel), 0.67 (SemEval), and 0.81 (MEN). We should note that these datasets are not specific to social media and even so, our trained embeddings outperform the standard pre-trained fastText in two datasets. In particular, there seems to be a marked difference between similarity and relatedness, where our Twitter embeddings appear to be more suited to relatedness.

Tweet Embedding Results. For tweet embeddings we explore a tweet retrieval task setting which consists of finding the reply to a given tweet from the 10k replies in the search space. We randomly sampled 3k tweet-reply pairs that do not overlap with training data and split them into 3 sets of 1k pairs. We report accuracy@1 and average models' performance on the 3 sets. We also include results on sentence similarity, using the STS-benchmark (Cer et al., 2017) and reporting Spearman's correlation. We list tweet-reply retrieval accuracy and STS-benchmark Spearman's correlation in Table 7.10. We compare with recent supervised (Reimers and Gurevych, 2019, Sentence-BERT; `all-mpnet-base-v2`), and unsupervised (Liu et al., 2021, Mirror-BERT), (Gao et al., 2021, SimCSE) sentence embedding models. On the task of tweet-reply retrieval, our tweet-embeddings model significantly outperforms `all-mpnet-base-v2` trained with around 1B sentence pairs. This highlights the importance of in-domain training. On the STS-Benchmark,

`all-mpnet-base-v2` achieves the best performance and our tweet-embeddings perform the worst among baselines but they are generally in a similar ballpark. To complement this evaluation, we plan to test our tweet embeddings with a textual similarity dataset in the tweet domain in the future.

7.4 LMQG

Generating questions along with associated answers from a text has applications in several domains, such as creating reading comprehension tests for students, or improving document search by providing auxiliary questions and answers based on the query. Training models for QAG is not straightforward due to the expected structured output (i.e. a list of question and answer pairs), as it requires more than generating a single sentence. This results in a small number of publicly accessible QAG models. In this section, we introduce AutoQG, an online service for multilingual QAG, along with `lmqg`, an all-in-one Python package for model fine-tuning, generation, and evaluation. We also release QAG models in eight languages fine-tuned on a few variants of pre-trained EDLMs, which can be used online via AutoQG or locally via `lmqg`. With these resources, practitioners of any level can benefit from a toolkit that includes a web interface for end users, and easy-to-use code for developers who require custom models or fine-grained controls for generation.

7.4.1 Introduction

QAG is a text generation task seeking to output a list of question-answer pairs based on a given paragraph or sentence (i.e. the context). It has been used in many NLP applications, including unsupervised question answering modeling (Lewis et al., 2019; Zhang and Bansal, 2019; Puri et al., 2020), fact-checking (Ousidhoum et al., 2022), semantic role labeling (Pyatkin et al., 2021), and as an educational tool (Heilman and Smith, 2010; Lindberg et al., 2013). The most analysed setting in the literature, however, has been QG with pre-defined answers, as this simplifies the

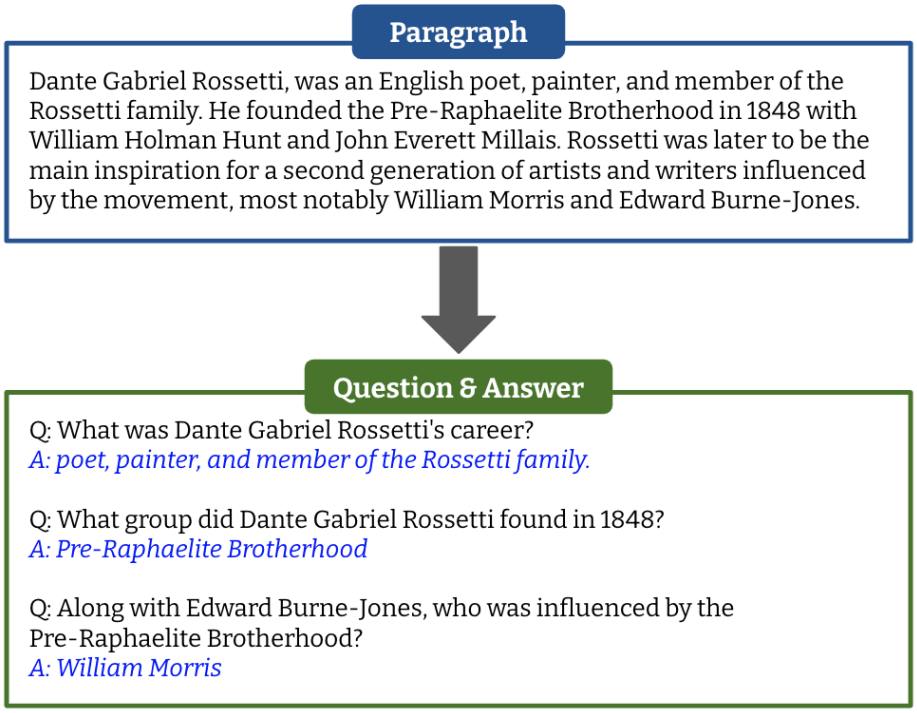

Figure 7.4: An example of QAG given a paragraph as context.

task and makes the evaluation more straightforward.

Despite its versatility, QAG remains a challenging task due to the difficulty of generating compositional outputs containing a list of question and answer pairs as shown in Figure 7.4, with recent works mainly relying on extended pipelines that include several ad-hoc models (Lewis et al., 2021; Bartolo et al., 2021). These works integrate QAG into their in-house software, preventing models to be publicly released, and their complex pipelines make them hard to reproduce and use by practitioners.

In this section, we introduce an open set of software tools and resources to assist on the development and employment of QAG models for different types of users. We

publicly release the following resources:[27]

- `lmqg`,[28] a Python package for QAG model fine-tuning and inference on EDLMs, as well as evaluation scripts, and a deployment API hosting QAG models for developers.
- 16 models for English, and three diverse models for each of the seven languages integrated into our library, all fine-tuned on QG-Bench (Ushio et al., 2022a) and available on the HuggingFace.
- *AutoQG* (`https://autoqg.net`), a website where developers and end users can interact with our multilingual QAG models.

7.4.2 Models and Datasets

Our QAG toolkit makes use of pre-existing models and datasets, fully compatible with the HuggingFace. This makes our library easily extendable in the future as newer datasets and better models emerge. In this section, we describe the datasets (§ 7.4.2.1) and models (§ 7.4.2.2) currently available through `lmqg` and AutoQG.

7.4.2.1 Multilingual Datasets

Our toolkit integrates all QG datasets available in QG-Bench (Ushio et al., 2022a). QG-Bench is a multilingual QG benchmark consisting of a suite of unified QG datasets in different languages. In particular, we integrate the following datasets:

[27] All the resources except for the datasets are released under an open MIT license, while the datasets follow the license of their original release.

[28] `https://github.com/asahi417/lm-question-generation`

- SQuAD (English) (Rajpurkar et al., 2016) (English)
- SQuADShifts (Miller et al., 2020) (English)
- SubjQA (Bjerva et al., 2020) (English)
- JAQuAD (So et al., 2022) (Japanese)
- GerQuAD (Möller et al., 2021) (German)
- SberQuAd (Efimov et al., 2020) (Russian)
- KorQuAD (Lim et al., 2019) (Korean)
- FQuAD (d'Hoffschmidt et al., 2020) (French)
- Spanish SQuAD (Casimiro Pio et al., 2019) (Spanish)
- Italian SQuAD (Croce et al., 2018) (Italian).

QG-Bench is available through our official `lmqg` HuggingFace project page and GitHub[29].

7.4.2.2 Models

Aiming to make QAG models publicly accessible in several languages, we used `lmqg` to fine-tune LMs using QG-Bench (§ 7.4.2.1). First, we defined a pipeline QAG model architecture consisting of two independent models: one for Answer Extraction (AE) and one for QG. During training, the AE model learns to find an answer in each sentence of a given paragraph, while the QG model learns to generate

[29] https://github.com/asahi417/lm-question-generation/blob/master/QG_BENCH.md

a question given an answer from a paragraph. To generate question-answer pairs at generation time, the AE model first extracts answers from all the sentences in a given paragraph, and then these are used by the QG model to generate a question for each answer. While not directly evaluated in this paper, we also integrated other types of QAG methods such as multitask and end2end QAG (Ushio et al., 2023), all available via the `lmqg` library (§ 7.4.3) as well as AutoQG (§ 7.4.5).

As pre-trained LMs, we integrated T5, Flan-T5, and BART for English; and mT5 (Xue et al., 2021) and mBART (Liu et al., 2020) for non-English QAG models. The pre-trained weights were taken from checkpoints available in the HuggingFace Hub as below:

- `t5-{small,base,large}`
- `google/flan-t5-{small,base,large}`
- `facebook/bart-{base,large}`
- `google/mt5-{small,base}`
- `facebook/mbart-large-cc25`

All the fine-tuned QAG models are publicly available in our official HuggingFace Hub. While we initially integrated these models, users can easily fine-tune others using `lmqg`, as we show in § 7.4.3.

7.4.3 lmqg: An All-in-one QAG Toolkit

In this section, we introduce `lmqg` (**L**anguage **M**odel for **Q**uestion **G**eneration), a Python library for fine-tuning LMs on QAG (§ 7.4.3.1), generating question-answer pairs (§ 7.4.3.2), and evaluating QAG models (§ 7.4.3.3). Additionally, with `lmqg`, we build a REST API to host QAG models to generate question and answer interactively (§ 7.4.5). `lmqg` is inter-operable with the HuggingFace ecosystem, as it can directly make use of the datasets and models already shared on the HuggingFace Hub.

7.4.3.1 QAG Model Fine-tuning

Fine-tuning is performed via `GridSearcher`, a class to run EDLM fine-tuning with hyper-parameter optimization. It performs LM fine-tuning with a two-stage optimization of hyper-parameter, a set of parameters to be used at fine-tuning such as learning rate or batch size. Let us assume that we want to find an optimal combination of the learning rate and random seed from a list of candidates [1e-4,1e-5] and [0,1] for learning rate and random seed respectively on QG as an example. We also assume a training and a validation dataset to train a model on the task and an evaluation score that reflects a performance of a model (eg. BLEU4(Papineni et al., 2002)), and we define a search-space as a set including all the combinations of those candidates, i.e. {(1e-4, 0), (1e-4, 1), (1e-5, 0), (1e-5, 1)}. The goal of the `GridSearcher` is to find the best combination to train a model on the training dataset for the target task over the search-space with respect to the evaluation score computed on the validation dataset.

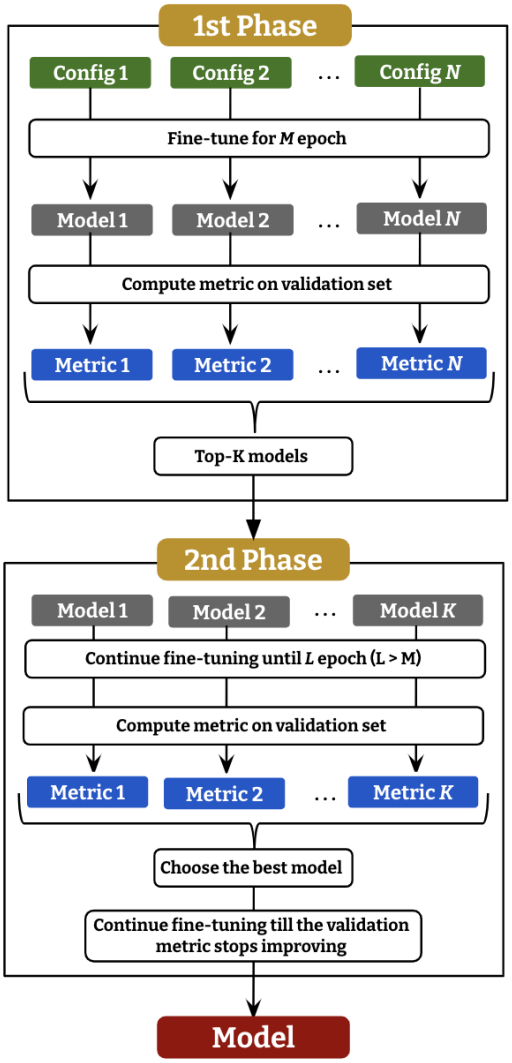

Figure 7.5: An overview of the hyper-parameter search implemented as `GridSearcher`.

Brute-force approach such as to train model over every combination in the search-space can be a highly-inefficient, so `GridSearcher` employs a two-stage search method to avoid training for all the combinations, while being able to reach to the optimal combination as possible. To be precise, given an epoch size L (`epoch`), the first stage fine-tunes all the combinations over the search-space, and pauses fine-tuning

at epoch M (`epoch_partial`). The top-K combinations (`n_max_config`) are then selected based on the evaluation score computed over the validation dataset, and they are resumed to be fine-tuned until the last epoch. Once the K chosen models are fine-tuned at second stage, the best model is selected based on the evaluation score, which is kept being fine-tuned until the evaluation score decreases.

For example, the following code shows how we can fine-tune T5 (Raffel et al., 2020) on SQuAD (Rajpurkar et al., 2016), with the QAG model explained in § 7.4.2.2. Since we decomposed QAG into AE and QG, two models need to be fine-tuned independently.

```
from lmqg import GridSearcher

# instantiate AE trainer
trainer_ae = GridSearcher(
dataset_path="lmqg/qg_squad",
input_types="paragraph_sentence",
output_types="answer",
model="t5-large")

# train AE model
trainer_ae.train()

# instantiate QG trainer
trainer_qg = GridSearcher(
dataset_path="lmqg/qg_squad",
input_types="paragraph_answer",
output_types="question",
```

```
    model="t5-large")

    # train QG model
    trainer_qg.train()
```

The corresponding dataset, `lmqg/qg_squad`,[30] has as columns:

- `paragraph_answer`: answer-highlighted paragraph
- `paragraph_sentence`: sentence-highlighted paragraph
- `question`: target question
- `answer`: target answer

The input and the output to the QG model are `paragraph_answer` and `question`, while those to the AE model are `paragraph_sentence` and `answer`. The inputs and the outputs can be specified by passing the name of each column in the dataset to the arguments, `input_types` and `output_types` when instantiating `GridSearcher`.

7.4.3.2 QAG Model Generation

In order to generate question-answer pairs from a fine-tuned QAG model, `lmqg` provides the `TransformersQG` class. It takes as input a path to a local model checkpoint or a model name on the HuggingFace Hub in order to generate predictions in a single line of code. The following code snippet shows how to generate a list of question and answer pairs with the fine-tuned QAG model presented in § 7.4.2.2.

[30] https://huggingface.co/datasets/lmqg/qg_squad

`TransformersQG` decides which model to use for each of AE and QG based on the arguments `model_ae` and `model`.

```
from lmqg import TransformersQG

# instantiate model
model = TransformersQG(
model="lmqg/t5-base-squad-qg",
model_ae="lmqg/t5-base-squad-ae"
)

# input paragraph
x = """William Turner was an English
painter who specialised in watercolour
landscapes. One of his best known
pictures is a view of the city of
Oxford from Hinksey Hill."""

# generation
model.generate_qa(x)
[
(
"Who was an English painter
 specialised in watercolour
 landscapes?",
"William Turner"
),
(
```

```
"Where is William Turner's
 view of Oxford?",
"Hinksey Hill."
)
]
```

7.4.3.3 QAG Model Evaluation

Similar to other text-to-text generation tasks, we implement an evaluation mechanism that compares the set of generated question-answer pairs $\tilde{\mathcal{Q}}_p = \{(\tilde{q}^1, \tilde{a}^1), (\tilde{q}^2, \tilde{a}^2), \dots\}$ to a reference set of gold question-answer pairs $\mathcal{Q}_p = \{(q^1, a^1), (q^2, a^2), \dots\}$ given an input paragraph p. Let us define a function to evaluate a single question-answer pair to its reference pair as

$$d_{q,a,\tilde{q},\tilde{a}} = s\big(t(q,a), t(\tilde{q},\tilde{a})\big) \tag{7.5}$$

$$t(q,a) = \text{“}\texttt{question:}\{q\}, \texttt{answer:}\{a\}\text{”} \tag{7.6}$$

where s is a reference-based metric, and we compute the F_1 score as the final metric as below:

$$F_1 = 2\frac{R \cdot P}{R + P} \tag{7.7}$$

$$R = \text{mean}\Big(\Big[\max_{(q,a)\in\mathcal{Q}_c}\big(d_{q,a,\tilde{q},\tilde{a}}\big)\Big]_{(\tilde{q},\tilde{a})\in\tilde{\mathcal{Q}}_c}\Big) \tag{7.8}$$

$$P = \text{mean}\Big(\Big[\max_{(\tilde{q},\tilde{a})\in\tilde{\mathcal{Q}}_c}\big(d_{q,a,\tilde{q},\tilde{a}}\big)\Big]_{(q,a)\in\mathcal{Q}_c}\Big) \tag{7.9}$$

Conceptually, the recall (7.8) and precision (7.9) computations attempt to “align” each generated question-answer pair to its “most relevant” reference pair. As with

traditional precision and recall metrics, precision is aimed at evaluating whether the predicted question-answer pairs are *correct* (or in this case, aligned with the reference question-answer pairs), and recall tests whether there are enough high-quality question-answer pairs. Thus, we refer to the score in (7.7) as the **QAAligned F1 score**. The quality of the alignment directly depends on the underlying metric s. Furthermore, the complexity of QAAligned is no more than the complexity of the underlying metric, and invariant to the order of generated pairs because of the alignment at computing recall and precision.

Out-of-the-box, `lmqg` implements two variants based on the choice of `base_metric`s (used for evaluation in § 7.4.4.1): QAAligned BS using BERTScore (Zhang et al., 2019b) and QAAligned MS using MoverScore (Zhao et al., 2019). We selected these two metrics as they correlate well with human judgements in QG (Ushio et al., 2022a). Nevertheless, the choice of `base_metric` is flexible and users can employ other NLG evaluation metrics such as BLEU4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), or ROUGE$_{\text{L}}$ (Lin, 2004). With `lmqg`, QAAligned score can be computed as shown in the code snippet below:

```
from lmqg import QAAlignedF1Score

# gold reference and generation
ref = [
"question: What makes X?, answer: Y",
"question: Who made X?, answer: Y"]
pred = [
"question: What makes X?, answer: Y",
"question: Who build X?, answer: Y",
"question: When X occurs?, answer: Y"]
```

```
# compute QAAligned BS
scorer = QAAlignedF1Score(
base_metric="bertscore")
scorer.get_score(pred, ref)

# compute QAAligned MS
scorer = QAAlignedF1Score(
base_metric="moverscore"
)
scorer.get_score(pred, ref)
```

7.4.4 Evaluation

In this section, we provide experimental results of the default models integrated into LMQG.

7.4.4.1 Experimental Settings

We rely on the QAG models and datasets included in the library (see § 7.4.2). The individual QG components of each model (i.e. the generation of a question given an answer in a paragraph) were extensively evaluated in Ushio et al. (2022a). For this evaluation, therefore, we focus on the quality of the predicted questions and answers given a paragraph (i.e. the specific answer is not pre-defined). For each model, we fine-tune, make predictions and compute their QAAligned scores via `lmqg`.

Model	QAAligned BS	QAAligned MS
$\text{BART}_{\text{BASE}}$	92.8 / 93.0 / 92.8	64.2 / 64.1 / 64.5
$\text{BART}_{\text{LARGE}}$	**93.2** / **93.4** / **93.1**	**64.8** / 64.6 / **65.0**
T5_{SMALL}	92.3 / 92.5 / 92.1	63.8 / 63.8 / 63.9
T5_{BASE}	92.8 / 92.9 / 92.6	64.4 / 64.4 / 64.5
T5_{LARGE}	93.0 / 93.1 / 92.8	64.7 / **64.7** / 64.9
$\text{Flan-T5}_{\text{SMALL}}$	92.3 / 92.1 / 92.5	63.8 / 63.8 / 63.8
$\text{Flan-T5}_{\text{BASE}}$	92.6 / 92.5 / 92.8	64.3 / 64.4 / 64.3
$\text{Flan-T5}_{\text{LARGE}}$	92.7 / 92.6 / 92.9	64.6 / **64.7** / 64.5

Table 7.11: QAAligned scores ($F_1/P/R$) on the test set of SQuAD dataset by different QAG models, where the best score in each metric is shown in boldface.

7.4.4.2 Experimental Results

Monolingual Results (English). Table 7.11 presents the test results on SQuAD for seven English models based on BART, T5 and Flan-T5. The QAG model based on $\text{BART}_{\text{LARGE}}$ proves to be the best aligned with gold reference question and answers among most of the metrics. As with other QG experiments and NLP in general, the larger models prove more reliable.

Multilingual Results. Table 7.12 shows the test results of three multilingual models (mBART, $\text{mT5}_{\text{SMALL}}$ and mT5_{BASE}) in seven languages other than English, using their corresponding language-specific SQuAD-like datasets in QG-Bench for fine-tuning and evaluation.[31] In this evaluation, no single LM produces the best results across the board, yet QAG models based on $\text{mT5}_{\text{SMALL}}$ and mT5_{BASE} are generally better than those based on mBART.

[31]The result of mBART in German is zero. Upon further inspection, we found that the fine-tuned answer extraction module did not learn properly, probably due to the limited size of the German dataset. T5 models, however, proved more reliable in this case.

	Language	QAAligned BS	QAAligned MS
$\text{mT5}_{\text{SMALL}}$	German	**81.2** / **80.0** / **82.5**	**54.3** / **54.0** / **54.6**
	Spanish	79.9 / 77.5 / 82.6	54.8 / 53.3 / 56.5
	French	**79.7** / **77.6** / **82.1**	**53.9** / **52.7** / **55.3**
	Italian	81.6 / 81.0 / **82.3**	**55.9** / 55.6 / **56.1**
	Japanese	79.8 / 76.8 / 83.1	55.9 / 53.8 / 58.2
	Korean	80.5 / 77.6 / 83.8	**83.0** / **79.4** / **87.0**
	Russian	77.0 / 73.4 / 81.1	55.5 / 53.2 / 58.3
mT5_{BASE}	German	76.9 / 76.3 / 77.6	53.0 / 52.9 / 53.1
	Spanish	**80.8** / **78.5** / **83.3**	**55.3** / **53.7** / **57.0**
	French	68.6 / 67.6 / 69.7	47.9 / 47.4 / 48.4
	Italian	**81.7** / **81.3** / 82.2	55.8 / **55.7** / 56.0
	Japanese	**80.3** / **77.1** / **83.9**	**56.4** / **54.0** / **59.1**
	Korean	77.3 / 76.4 / 78.3	77.5 / 76.3 / 79.0
	Russian	77.0 / 73.4 / 81.2	55.6 / 53.3 / 58.4
mBART	German	0 / 0 / 0	0 / 0 / 0
	Spanish	79.3 / 76.8 / 82.0	54.7 / 53.2 / 56.4
	French	75.6 / 74.0 / 77.2	51.8 / 51.0 / 52.5
	Italian	40.1 / 40.4 / 39.9	27.8 / 28.1 / 27.5
	Japanese	76.7 / 74.8 / 78.9	53.6 / 52.3 / 55.1
	Korean	**80.6** / **77.7** / **84.0**	82.7 / 79.0 / **87.0**
	Russian	**79.1** / **75.9** / **82.9**	**56.3** / **54.0** / **58.9**

Table 7.12: QAAligned scores ($F_1/P/R$) on the test set of QG-Bench by different QAG models, where the best score in each language is shown in boldface.

7.4.5 AutoQG

Finally, we present AutoQG (`https://autoqg.net`), an online QAG demo where users can generate question-answer pairs for texts in eight languages (English, German, Spanish, French, Italian, Japanese, Korean, Russian) by simply providing a context document. We deploy the QAG models described in § 7.4.2. In addition to the features described above, the online demo shows perplexity computed via `lmppl`,[32] a Python library to compute perplexity given any LM architecture. This feature helps us provide a ranked list of generation to the user. Although we can compute perplexity for non-English generations based on the QAG models in

[32] `https://pypi.org/project/lmppl`

each language, it entails large memory requirements on the the hosting server. As such, we compute a lexical overlap between the question and the document as a computationally-light alternative to the perplexity, which is defined as:

$$1 - \frac{|q \cap p|}{|q|} \tag{7.10}$$

where $|\cdot|$ is the number of characters in a string, and $q \cap p$ is the longest sub-string of the question q matched to the paragraph p.

Figure 7.6 and Figure 7.7 show examples of the interface with English and Japanese QAG, where there is a tab to select QAG models, language, and parameters at generation including the beam size and the value for nucleus sampling (Holtzman et al., 2020). Optionally, users can specify an answer and generate a single question on it with the QG model, as shown in Figure 7.8. A short introduction video to AutoQG is available at `https://youtu.be/T6G-D9JtYyc`.

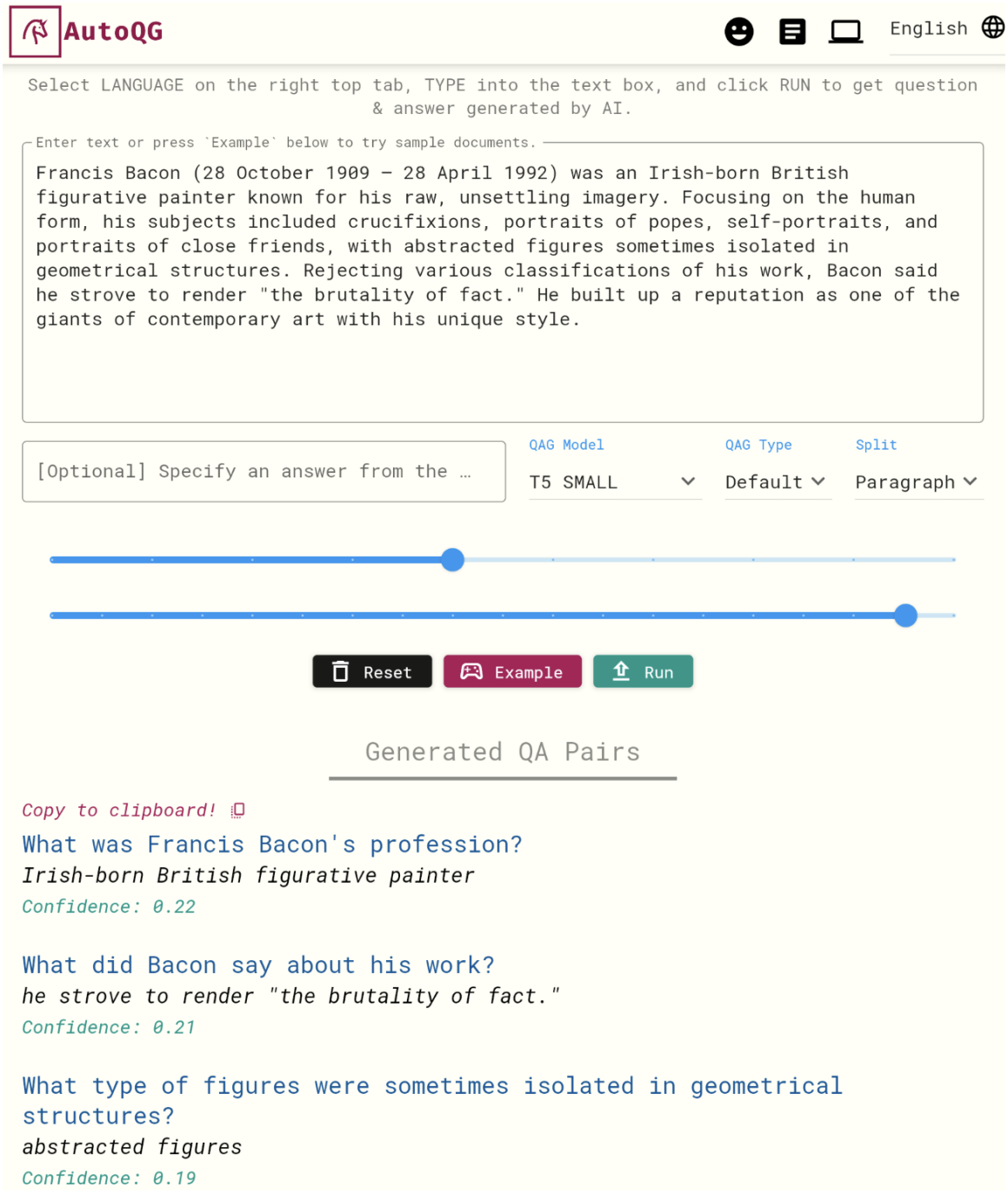

Figure 7.6: A screenshot of AutoQG with an example of question and answer generation over a paragraph.

Figure 7.7: A screenshot of AutoQG with an example of question and answer generation over a paragraph in Japanese.

Figure 7.8: A screenshot of AutoQG when an answer is specified by the user.

Bibliography

Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. 2019. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*.

Oshin Agarwal, Yinfei Yang, Byron C Wallace, and Ani Nenkova. 2021. Entity-switched datasets: An approach to auditing the in-domain robustness of named entity recognition models. *arXiv preprint arXiv:2004.04123*.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Boulder, Colorado. Association for Computational Linguistics.

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pre-trained language models. In *Proceedings of the 58th Annual Meeting of the As-*

sociation for Computational Linguistics, pages 7747–7763, Online. Association for Computational Linguistics.

Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of COLING*, pages 1638–1649.

Dimitrios Alivanistos, Selene Báez Santamaría, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as probing: Using language models for knowledge base construction. *arXiv preprint arXiv:2208.11057*.

Carl Allen and Timothy Hospedales. 2019. Analogies explained: Towards understanding word embeddings. In *International Conference on Machine Learning*, pages 223–231.

Dimosthenis Antypas, Asahi Ushio, Jose Camacho-Collados, Vitor Silva, Leonardo Neves, and Francesco Barbieri. 2022. Twitter topic classification. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3386–3400, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Mercedes Arguello Casteleiro, Julio Des Diz, Nava Maroto, Maria Jesus Fernandez Prieto, Simon Peters, Chris Wroe, Carlos Sevillano Torrado, Diego Maseda Fernandez, and Robert Stevens. 2020. Semantic deep learning: Prior knowledge and a type of four-term embedding analogy to acquire treatments for well-known diseases. *JMIR Med Inform*, 8(8).

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Ignacio Arroyo-Fernández, Carlos-Francisco Méndez-Cruz, Gerardo Sierra, Juan-Manuel Torres-Moreno, and Grigori Sidorov. 2019. Unsupervised sentence representations as word information series: Revisiting tf–idf. *Computer Speech & Language*, 56:107–129.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Kevin D Ashley. 1988. Arguing by analogy in law: A case-based model. In *Analogical Reasoning*, pages 205–224. Springer.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364, Nagoya, Japan. Asian Federation of Natural Language Processing.

Timothy Baldwin, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-

generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135, Beijing, China. Association for Computational Linguistics.

Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. 2016. Overview of the evalita 2016 sentiment polarity classification task. In *Proceedings of third Italian conference on computational linguistics (CLiC-it 2016) & fifth evaluation campaign of natural language processing and speech tools for Italian. Final Workshop (EVALITA 2016)*.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa-Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. SemEval 2018 task 2: Multilingual emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 24–33, New Orleans, Louisiana. Association for Computational Linguistics.

Francesco Barbieri, Luis Espinosa Anke, and Jose Camacho-Collados. 2022. Xlm-t: Multilingual language models in twitter for sentiment analysis and beyond. In *Proceedings of the Language Resources and Evaluation Conference*, pages 258–266, Marseille, France. European Language Resources Association.

Nelly Barbot, Laurent Miclet, and Henri Prade. 2019. Analogy Between Concepts. *Artificial Intelligence*, 275:487–539.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, Edinburgh, UK. Association for Computational Linguistics.

Max Bartolo, Tristan Thrush, Robin Jia, Sebastian Riedel, Pontus Stenetorp, and Douwe Kiela. 2021. Improving question answering model robustness with synthetic adversarial data generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8830–8848, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Slobodan Beliga, Ana Meštrović, and Sanda Martinčić-Ipšić. 2015. An overview of graph-based keyword extraction methods and approaches. *Journal of information and organizational sciences*, 39(1):1–20.

Farah Benamara, Cyril Grouin, Jihen Karoui, Véronique Moriceau, and Isabelle Robba. 2017. Analyse d'opinion et langage figuratif dans des tweets: présentation et résultats du défi fouille de textes deft2017. In *Défi Fouille de Textes DEFT2017. Atelier TALN 2017*. Association pour le Traitement Automatique des Langues (ATALA).

Bhavya, Jinjun Xiong, and Chengxiang Zhai. 2023. CAM: A large language model-based creative analogy mining framework. In *Proceedings of the ACM Web Con-*

ference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023, pages 3903–3914. ACM.

Bhavya Bhavya, Jinjun Xiong, and ChengXiang Zhai. 2022. Analogy generation by prompting large language models: A case study of instructgpt. In *Proceedings of the 15th International Conference on Natural Language Generation*, pages 298–312, Waterville, Maine, USA and virtual meeting. Association for Computational Linguistics.

Mokhtar-Boumeyden Billami, José Camacho-Collados, Evelyne Jacquey, and Laurence Kister. 2014. Annotation sémantique et validation terminologique en texte intégral en SHS. In *Proceedings of TALN*, pages 363–376.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Johannes Bjerva, Nikita Bhutani, Behzad Golshan, Wang-Chiew Tan, and Isabelle Augenstein. 2020. SubjQA: A Dataset for Subjectivity and Review Comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5480–5494, Online. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. Interpreting pretrained contextualized representations via reductions to static embeddings. In *Proceed-*

ings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4758–4781.

Rajesh Bordawekar and Oded Shmueli. 2017. Using word embedding to enable semantic queries in relational databases. In *Proceedings of the 1st Workshop on Data Management for End-to-End Machine Learning*, pages 1–4.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar. Association for Computational Linguistics.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Adrian Boteanu and Sonia Chernova. 2015. Solving and explaining analogy questions using semantic networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 667–672, New Orleans, Louisiana. Association for Computational Linguistics.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-

based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*, pages 543–551.

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing Relational Knowledge from BERT. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7456–7463.

Zied Bouraoui, Shoaib Jameel, and Steven Schockaert. 2018. Relation induction in word embeddings revisited. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1627–1637, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zied Bouraoui and Steven Schockaert. 2019. Automated rule base completion as bayesian concept induction. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pages 6228–6235.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learn-

ers. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Henrico Brum and Maria das Graças Volpe Nunes. 2018. Building a sentiment corpus of tweets in Brazilian Portuguese. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, 49(1-47).

Jose Camacho-Collados, Yerai Doval, Eugenio Martínez-Cámara, Luis Espinosa-Anke, Francesco Barbieri, and Steven Schockaert. 2020. Learning cross-lingual word embeddings from twitter via distant supervision. In *Proceedings of the international AAAI conference on web and social media*, volume 14, pages 72–82.

Jose Camacho-Collados, Luis Espinosa-Anke, Jameel Shoaib, and Steven Schockaert. 2019. A latent variable model for learning distributional relation vectors. In *Proceedings of IJCAI*.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. SemEval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26, Vancouver, Canada. Association for Computational Linguistics.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II 17*, pages 160–172. Springer.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alipio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Carrino Casimiro Pio, Costa-jussa Marta R., and Fonollosa Jose A. R. 2019. Automatic Spanish Translation of the SQuAD Dataset for Multilingual Question Answering. *arXiv e-prints*, page arXiv:1912.05200v1.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. 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, Vancouver, Canada. Association for Computational Linguistics.

Jiangjie Chen, Rui Xu, Ziquan Fu, Wei Shi, Zhongqiao Li, Xinbo Zhang, Changzhi Sun, Lei Li, Yanghua Xiao, and Hao Zhou. 2022a. E-KAR: A benchmark for rationalizing natural language analogical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3941–3955, Dublin, Ireland. Association for Computational Linguistics.

Jiaoyan Chen, Yuan He, Ernesto Jiménez-Ruiz, Hang Dong, and Ian Horrocks. 2022b. Contextual semantic embeddings for ontology subsumption prediction. *CoRR*, abs/2202.09791.

Zhanwen Chen, Saed Rezayi, and Sheng Li. 2023. More knowledge, less bias: Unbiasing scene graph generation with explicit ontological adjustment. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2023, Waikoloa, HI, USA, January 2-7, 2023*, pages 4012–4021. IEEE.

Hsiao-Yu Chiang, Jose Camacho-Collados, and Zachary Pardos. 2020. Understanding the source of semantic regularities in word embeddings. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 119–131, Online. Association for Computational Linguistics.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Mark Cieliebak, Jan Milan Deriu, Dominic Egger, and Fatih Uzdilli. 2017. A Twitter corpus and benchmark resources for German sentiment analysis. In *Proceedings of the Fifth International Workshop on Natural Language Processing for So-*

cial Media, pages 45–51, Valencia, Spain. Association for Computational Linguistics.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. 2023. Crawling the internal knowledge-base of language models. *arXiv preprint arXiv:2301.12810*.

Nigel Collier, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 73–78, Geneva, Switzerland. COLING.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning

at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6022–6034, Online. Association for Computational Linguistics.

Danilo Croce, Alexandra Zelenanska, and Roberto Basili. 2018. Neural learning for question answering in italian. In *AI*IA 2018 – Advances in Artificial Intelligence*, pages 389–402, Cham. Springer International Publishing.

Tamara Czinczoll, Helen Yannakoudakis, Pushkar Mishra, and Ekaterina Shutova. 2022. Scientific and creative analogies in pretrained language models. *arXiv preprint arXiv:2211.15268*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Joe Davison, Joshua Feldman, and Alexander Rush. 2019a. Commonsense knowledge mining from pretrained models. 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 1173–1178, Hong Kong, China. Association for Computational Linguistics.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019b. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1173–1178.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.*, 41(6):391–407.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 198–206, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. *arXiv preprint arXiv:2003.07892*.

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

Martin d'Hoffschmidt, Wacim Belblidia, Quentin Heinrich, Tom Brendlé, and Maxime Vidal. 2020. FQuAD: French question answering dataset. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1193–1208, Online. Association for Computational Linguistics.

Patrick Drouin. 2003. Term extraction using non-technical corpora as a point of leverage. *Terminology*, 9(1):99–115.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3519–3530, Osaka, Japan. The COLING 2016 Organizing Committee.

Manuel C. Díaz-Galiano, Eugenio Martínez-Cámara, M. Ángel García Cumbreras, Manuel García Vega, and Julio Villena Román. 2018. The democratization of deep learning in tass 2017. *Procesamiento del Lenguaje Natural*, 60:37–44.

Pavel Efimov, Andrey Chertok, Leonid Boytsov, and Pavel Braslavski. 2020. Sberquad–russian reading comprehension dataset: Description and analysis. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 3–15. Springer.

Samhaa R El-Beltagy and Ahmed Rafea. 2009. Kp-miner: A keyphrase extraction system for english and arabic documents. *Information systems*, 34(1):132–144.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Luis Espinosa-Anke and Steven Schockaert. 2018. SeVeN: Augmenting word embeddings with unsupervised relation vectors. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2653–2665, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 55–65.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262.

Jerome A. Feldman and Dana H. Ballard. 1982. Connectionist models and their properties. *Cognitive Science*, 6(3):205–254.

Raul Castro Fernandez, Essam Mansour, Abdulhakim A Qahtan, Ahmed Elmagarmid, Ihab Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2018. Seeping semantics: Linking datasets using word embeddings for

data discovery. In *2018 IEEE 34th International Conference on Data Engineering*, pages 989–1000.

Lev Finkelstein, Gabrilovich Evgeniy, Matias Yossi, Rivlin Ehud, Solan Zach, Wolfman Gadi, and Ruppin Eytan. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115.

Maxwell Forbes, Ari Holtzman, and Yejin Choi. 2019. Do neural language representations learn physical commonsense? In *Proceedings of the 41th Annual Meeting of the Cognitive Science Society*, pages 1753–1759.

Andreas Fürst, Elisabeth Rumetshofer, Viet Tran, Hubert Ramsauer, Fei Tang, Johannes Lehner, David Kreil, Michael Kopp, Günter Klambauer, Angela Bitto-Nemling, et al. 2021. Cloob: Modern hopfield networks with infoloob outperform clip. *arXiv preprint arXiv:2110.11316*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland

Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Dedre Gentner and Arthur B Markman. 1997. Structure mapping in analogy and similarity. *American psychologist*, 52(1):45.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Alex Gittens, Dimitris Achlioptas, and Michael W Mahoney. 2017. Skip-gram-zipf+ uniform= vector additivity. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 69–76.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15, San Diego, California. Association for Computational Linguistics.

Ashok Goel. 2019. Computational design, analogy, and creativity. In *Computational Creativity*, pages 141–158. Springer.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th international conference on World wide web*, pages 661–670.

Jiuxiang Gu, Handong Zhao, Zhe Lin, Sheng Li, Jianfei Cai, and Mingyang Ling. 2019. Scene graph generation with external knowledge and image reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 1969–1978. Computer Vision Foundation / IEEE.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 368–378, Portland, Oregon, USA. Association for Computational Linguistics.

Shibo Hao, Bowen Tan, Kaiwen Tang, Hengzhe Zhang, Eric P Xing, and Zhiting Hu. 2022. Bertnet: Harvesting knowledge graphs from pretrained language models. *arXiv preprint arXiv:2206.14268*.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 415–422.

Hiroaki Hayashi, Zecong Hu, Chenyan Xiong, and Graham Neubig. 2020. Latent relation language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7911–7918.

Yuan He, Jiaoyan Chen, Denvar Antonyrajah, and Ian Horrocks. 2022. Bertmap: A bert-based ontology alignment system. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 5684–5691. AAAI Press.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California. Association for Computational Linguistics.

Benjamin Heinzerling and Kentaro Inui. 2021. 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, Online. Association for Computational Linguistics.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4129–4138.

Geoffrey E. Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Geoffrey E. Hinton, James L. McClelland, and David E. Rumelhart. 1986. Distributed representations. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1*, pages 77–109.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35:30016–30030.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Keith J Holyoak, Keith James Holyoak, and Paul Thagard. 1996. *Mental leaps: Analogy in creative thought*. MIT press.

Tom Hope, Joel Chan, Aniket Kittur, and Dafna Shahaf. 2017. Accelerating innovation through analogy mining. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 235–243.

John J. Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.

Danqi Hu, Charles M Jones, Valerie Zhang, and Xiaoyan Zhang. 2021. The rise of reddit: How social media affects retail investors and short-sellers' roles in price discovery. *Available at SSRN 3807655*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020a. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning (ICML)*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020b. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *arXiv preprint arXiv:2003.11080*.

Jie Huang, Kerui Zhu, Kevin Chen-Chuan Chang, Jinjun Xiong, and Wen-mei Hwu. 2022. DEER: Descriptive knowledge graph for explaining entity relationships. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6686–6698, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nicolas Hug, Henri Prade, Gilles Richard, and Mathieu Serrurier. 2016. Analogical classifiers: a theoretical perspective. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 689–697.

Eyke Hüllermeier. 2020. Towards analogy-based explanations in machine learning. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 205–217.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Dániel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. 2022. Opt-iml: Scaling language model instruction meta learning through the lens of generalization.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th*

Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 874–880, Online. Association for Computational Linguistics.

Siham Jabri, Azzeddine Dahbi, Taoufiq Gadi, and Abdelhak Bassir. 2018. Ranking of text documents using tf-idf weighting and association rules mining. In *2018 4th International Conference on Optimization and Applications (ICOA)*, pages 1–6. IEEE.

Parastoo Jafarzadeh, Zahra Amirmahani, and Faezeh Ensan. 2022. Learning to rank knowledge subgraph nodes for entity retrieval. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2519–2523. ACM.

Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. 2018a. Unsupervised learning of distributional relation vectors. In *Annual Meeting of the Association for Computational Linguistics*, pages 23–33.

Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. 2018b. Unsupervised learning of distributional relation vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Melbourne, Australia. Association for Computational Linguistics.

James Jardine and Simone Teufel. 2014. Topical pagerank: A model of scientific expertise for bibliographic search. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 501–510.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of*

the Association for Computational Linguistics, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Ji-Rong Wen, and Xin Zhao. 2022. *great truths are always simple* : a rather simple knowledge encoder for enhancing the commonsense reasoning capacity of pre-trained models. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1730–1741, Seattle, United States. Association for Computational Linguistics.

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

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Mandar Joshi, Eunsol Choi, Omer Levy, Daniel Weld, and Luke Zettlemoyer. 2019. pair2vec: Compositional word-pair embeddings for cross-sentence inference. 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 3597–3608, Minneapolis, Minnesota. Association for Computational Linguistics.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

David Jurgens, Saif Mohammad, Peter Turney, and Keith Holyoak. 2012. SemEval-2012 task 2: Measuring degrees of relational similarity. In **SEM 2012: The First*

Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012), pages 356–364, Montréal, Canada. Association for Computational Linguistics.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. Multilingual LAMA: Investigating knowledge in multilingual pretrained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3250–3258, Online. Association for Computational Linguistics.

Nora Kassner and Hinrich Schütze. 2020. 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, Online. Association for Computational Linguistics.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden. Association for Computational Linguistics.

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

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language*

Processing: System Demonstrations, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Pierre Lafon. 1980. Sur la variabilité de la fréquence des formes dans un corpus. *Mots. Les langages du politique*, 1(1):127–165.

Shibamouli Lahiri, Rada Mihalcea, and Po-Hsiang Lai. 2017. Keyword extraction from emails. *Nat. Lang. Eng.*, 23(2):295–317.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

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

Teven Le Scao and Alexander M Rush. 2021. How many data points is a prompt worth? *arXiv e-prints*, pages arXiv–2103.

Ludovic Lebart, A Salem, and Lisette Berry. 1998. *Exploring textual data*. Kluwer Academic Publishers.

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

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. 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, Online. Association for Computational Linguistics.

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4896–4910, Florence, Italy. Association for Computational Linguistics.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020b. MLQA: Evaluating cross-lingual extractive question answering. In *Pro-*

ceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7315–7330, Online. Association for Computational Linguistics.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Na Li, Zied Bouraoui, and Steven Schockaert. 2019a. Ontology completion using graph convolutional networks. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 435–452. Springer.

Shuyi Li, Shaojuan Wu, Xiaowang Zhang, and Zhiyong Feng. 2023. An analogical reasoning method based on multi-task learning with relational clustering. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, page 144–147, New York, NY, USA. Association for Computing Machinery.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2019b. Dice loss for data-imbalanced nlp tasks. *arXiv preprint arXiv:1911.02855*.

Davis Liang, Hila Gonen, Yuning Mao, Rui Hou, Naman Goyal, Marjan Ghazvininejad, Luke Zettlemoyer, and Madian Khabsa. 2023. Xlm-v: Overcom-

ing the vocabulary bottleneck in multilingual masked language models. *arXiv preprint arXiv:2301.10472*.

Seungyoung Lim, Myungji Kim, and Jooyoul Lee. 2019. Korquad1. 0: Korean qa dataset for machine reading comprehension. *arXiv preprint arXiv:1909.07005*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Dekang Lin and Patrick Pantel. 2001. DIRT - discovery of inference rules from text. In *Proceedings of the 7th Iternational Conference on Knowledge Discovery and Data Mining*, pages 323–328.

David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. 2013. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114, Sofia, Bulgaria. Association for Computational Linguistics.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 13–18, Berlin, Germany. Association for Computational Linguistics.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

Bing Liu and Ian Lane. 2017. Multi-domain adversarial learning for slot filling in spoken language understanding. *arXiv preprint arXiv:1711.11310*.

Fangyu Liu, Ivan Vulić, Anna Korhonen, and Nigel Collier. 2021. Fast, effective, and self-supervised: Transforming masked language models into universal lexical and sentence encoders. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1459, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fei Liu, Feifan Liu, and Yang Liu. 2010. A supervised framework for keyword extraction from meeting transcripts. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(3):538–548.

Weiyang Liu, Yan-Ming Zhang, Xingguo Li, Zhiding Yu, Bo Dai, Tuo Zhao, and Le Song. 2017. Deep Hyperspherical Learning. *Advances in neural information processing systems*, 30.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 257–266.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware lan-

guage modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971, Florence, Italy. Association for Computational Linguistics.

Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-collados. 2022. TimeLMs: Diachronic language models from Twitter. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 251–260, Dublin, Ireland. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of the Association for Computational Linguistics*, 4:231–244.

Samuel Marcos-Pablos and Francisco J García-Peñalvo. 2020. Information retrieval methodology for aiding scientific database search. *Soft Computing*, 24(8):5551–5560.

Luis Marujo, Márcio Viveiros, and João Paulo da Silva Neto. 2013. Keyphrase cloud generation of broadcast news. *arXiv preprint arXiv:1306.4606*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of NeurIPS*, volume 30.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1318–1327, Singapore. Association for Computational Linguistics.

Olena Medelyan and Ian H Witten. 2008. Domain-independent automatic keyphrase indexing with small training sets. *Journal of the American Society for Information Science and Technology*, 59(7):1026–1040.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Laurent Miclet, Sabri Bayoudh, and Arnaud Delhay. 2008. Analogical dissimilarity: definition, algorithms and two experiments in machine learning. *Journal of Artificial Intelligence Research*, 32:793–824.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. 2020. The effect of natural distribution shift on question answering models. In *International Conference on Machine Learning*, pages 6905–6916. PMLR.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. SemEval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17, New Orleans, Louisiana. Association for Computational Linguistics.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California. Association for Computational Linguistics.

Timo Möller, Julian Risch, and Malte Pietsch. 2021. Germanquad and germandpr: Improving non-english question answering and passage retrieval.

Silvia Necşulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 182–192, Denver, Colorado. Association for Computational Linguistics.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference*

on Empirical Methods in Natural Language Processing: System Demonstrations, pages 9–14, Online. Association for Computational Linguistics.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326. Springer.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, et al. 2018. Universal dependencies 2.2.

Farhad Nooralahzadeh, Jan Tore Lønning, and Lilja Øvrelid. 2019. Reinforcement-based denoising of distantly supervised ner with partial annotation. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 225–233.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*.

Nedjma Ousidhoum, Zhangdie Yuan, and Andreas Vlachos. 2022. Varifocal question generation for fact-checking. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2532–2544, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Jiaul H Paik. 2013. A novel tf-idf weighting scheme for effective ranking. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 343–352.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Denis Paperno and Marco Baroni. 2016. When the whole is less than the sum of its parts: How composition affects pmi values in distributional semantic vectors. *Computational Linguistics*, 42(2):345–350.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Zachary A. Pardos and Andrew J. H. Nam. 2020. A university map of course knowledge. *PLoS ONE*, 15(9).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

Braja Gopal Patra, Dipankar Das, Amitava Das, and Rajendra Prasath. 2015. Shared task on sentiment analysis in indian languages (sail) tweets-an overview. In *International Conference on Mining Intelligence and Knowledge Exploration*, pages 650–655. Springer.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand

Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011a. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011b. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. 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, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. 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, Hong Kong, China. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019a. 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, Hong Kong, China. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019b. 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.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020a. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *arXiv preprint arXiv:2005.00052*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2020. Embeddings in natural language processing: Theory and advances in vector representations of meaning. *Synthesis Lectures on Human Language Technologies*, 13(4):1–175.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, and Rada Mihalcea. 2020. Beneath the tip of the iceberg: Current challenges and new directions in sentiment analysis research. *IEEE Transactions on Affective Computing*.

Henri Prade and Gilles Richard. 2017. Analogical proportions and analogical reasoning-an introduction. In *International Conference on Case-Based Reasoning*, pages 16–32. Springer.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826, Online. Association for Computational Linguistics.

Valentina Pyatkin, Paul Roit, Julian Michael, Yoav Goldberg, Reut Tsarfaty, and Ido Dagan. 2021. Asking it all: Generating contextualized questions for any semantic role. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1429–1441, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Elie Raad and Joerg Evermann. 2015. The role of analogy in ontology alignment: A study on LISA. *Cognitive Systems Research*, 33:1–16.

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

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

Jack Rae, Geoffrey Irving, and Laura Weidinger. 2021. Language modelling at scale: Gopher, ethical considerations, and retrieval. *DeepMind Blog*.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. 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, Austin, Texas. Association for Computational Linguistics.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling

with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Kiamehr Rezaee and Jose Camacho-Collados. 2022. Probing relational knowledge in language models via word analogies. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3930–3936, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020*

Conference on Empirical Methods in Natural Language Processing, pages 5418–5426.

Eleanor Rosch. 1975. Cognitive representations of semantic categories. *Journal of experimental psychology: General*, 104(3):192.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1–20.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada. Association for Computational Linguistics.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Julio Cesar Salinas Alvarado, Karin Verspoor, and Timothy Baldwin. 2015. Domain adaption of named entity recognition to support credit risk assessment. In *Proceedings of the Australasian Language Technology Association Workshop 2015*, pages 84–90, Parramatta, Australia.

Enrico Santus, Anna Gladkova, Stefan Evert, and Alessandro Lenci. 2016a. The CogALex-V shared task on the corpus-based identification of semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 69–79, Osaka, Japan. The COLING 2016 Organizing Committee.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016b. Nine features in a random forest to learn taxonomical semantic relations. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4557–4564, Portorož, Slovenia. European Language Resources Association (ELRA).

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. EVALution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications*, pages 64–69, Beijing, China. Association for Computational Linguistics.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with SVCCA. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3257–3267.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. Sensembert: Context-enhanced sense embeddings for multilingual word sense disambiguation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8758–8765.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Confer-*

ence of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 255–269, Online. Association for Computational Linguistics.

Natalie Schluter. 2018. The word analogy testing caveat. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 242–246, New Orleans, Louisiana. Association for Computational Linguistics.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.

Alexander Thorsten Schutz et al. 2008. Keyphrase extraction from single documents in the open domain exploiting linguistic and statistical methods.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. 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, Online. Association for Computational Linguistics.

Vered Shwartz and Ido Dagan. 2016. Path-based vs. distributional information in recognizing lexical semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 24–29, Osaka, Japan. The COLING 2016 Organizing Committee.

Étienne Simon, Vincent Guigue, and Benjamin Piwowarski. 2019. Unsupervised information extraction: Regularizing discriminative approaches with relation distribution losses. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1378–1387.

ByungHoon So, Kyuhong Byun, Kyungwon Kang, and Seongjin Cho. 2022. Jaquad: Japanese question answering dataset for machine reading comprehension. *arXiv preprint arXiv:2202.01764*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936. PMLR.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Robyn Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3679–3686, Istanbul, Turkey. European Language Resources Association (ELRA).

Lucas Sterckx, Thomas Demeester, Johannes Deleu, and Chris Develder. 2015. Topical word importance for fast keyphrase extraction. In *Proceedings of the 24th International Conference on World Wide Web*, pages 121–122.

Miranda Stewart et al. 1999. *The Spanish language today*. Psychology Press.

Oren Sultan and Dafna Shahaf. 2022. Life is a circus and we are the clowns: Automatically finding analogies between situations and processes. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*,

pages 3547–3562, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Si Sun, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Jie Bao. 2020. Joint keyphrase chunking and salience ranking with bert. *arXiv preprint arXiv:2004.13639*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Yueqing Sun, Qi Shi, Le Qi, and Yu Zhang. 2022. JointLK: Joint reasoning with language models and knowledge graphs for commonsense question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5049–5060, Seattle, United States. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. *arXiv preprint arXiv:2103.11955*.

Zhong Tang, Wenqiang Li, and Yan Li. 2020. An improved term weighting scheme for text classification. *Concurrency and Computation: Practice and Experience*, 32(9):e5604.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. Ul2: Unifying language learning paradigms.

Wilson L. Taylor. 1953. Wordnet: a lexical database for english. *Journalism Bulletin*, 30(4):415–433.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *Proceeding of the 7th International Conference on Learning Representations (ICLR)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. 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.

Peter D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proc. of IJCAI*, pages 1136–1141.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Peter D Turney. 2008. The Latent Relation Mapping Engine: Algorithm and Experiments. *Journal of Artificial Intelligence Research*, 33:615–655.

Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor Shnayder. 2003. Combining independent modules in lexical multiple-choice problems. In *Recent Advances in Natural Language Processing III*, pages 101–110.

Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. 2022a. Generative language models for paragraph-level question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 670–688, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. 2023. An empirical comparison of lm-based question and answer generation methods. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics*, Toronto, Canada. Association for Computational Linguistics.

Asahi Ushio, Francesco Barbieri, Vitor Sousa, Leonardo Neves, and Jose Camacho-Collados. 2022b. Named entity recognition in Twitter: A dataset and analysis on short-term temporal shifts. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 309–319, Online only. Association for Computational Linguistics.

Asahi Ushio, Luis Espinosa Anke, Steven Schockaert, and Jose Camacho-Collados. 2021. BERT is to NLP what AlexNet is to CV: Can pre-trained language models identify analogies? 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 3609–3624, Online. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visulizing data using t-sne. *JMLR*, 9:2579 – 2605.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International*

Workshop on Semantic Evaluation, pages 39–50, New Orleans, Louisiana. Association for Computational Linguistics.

Marten van Schijndel, Aaron Mueller, and Tal Linzen. 2019. Quantity doesn't buy quality syntax with neural language models. 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 5831–5837, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*, volume 30.

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

Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2017. HyperLex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835.

Ivan Vulic, Edoardo Maria Ponti, Robert Litschko, Goran Glavas, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7222–7240.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1671–1682, Berlin, Germany. Association for Computational Linguistics.

Douglas Walton. 2010. Similarity, precedent and argument from analogy. *Artificial Intelligence and Law*, 18(3):217–246.

Xiaojun Wan and Jianguo Xiao. 2008a. Collabrank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 969–976.

Xiaojun Wan and Jianguo Xiao. 2008b. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860.

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a Markov random field language model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36, Minneapolis, Minnesota. Association for Computational Linguistics.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Chenguang Wang, Xiao Liu, and Dawn Song. 2020. Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*.

Chengyu Wang, Xiaofeng He, and Aoying Zhou. 2019a. SphereRE: Distinguishing lexical relations with hyperspherical relation embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1727–1737, Florence, Italy. Association for Computational Linguistics.

Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: deep knowledge-aware network for news recommendation. In *Proceedings of the 2018*

World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018, pages 1835–1844. ACM.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019b. KGAT: knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 950–958. ACM.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019c. Crossweigh: Training named entity tagger from imperfect annotations. 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 5157–5166.

Koki Washio and Tsuneaki Kato. 2018a. Filling missing paths: Modeling co-occurrences of word pairs and dependency paths for recognizing lexical semantic relations. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1123–1133.

Koki Washio and Tsuneaki Kato. 2018b. Neural latent relational analysis to capture lexical semantic relations in a vector space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 594–600, Brussels, Belgium. Association for Computational Linguistics.

Taylor Webb, Keith J Holyoak, and Hongjing Lu. 2022. Emergent analogical reasoning in large language models. *arXiv preprint arXiv:2212.09196*.

Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. 2015. Overview of the

biocreative v chemical disease relation (cdr) task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*, volume 14.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China. Association for Computational Linguistics.

Katrin Weller, Axel Bruns, Jean Burgess, Merja Mahrt, and Cornelius Puschmann. 2013. *Twitter and society*. Peter Lang New York.

Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2022. Symbolic knowledge distillation: from general language models to commonsense models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4602–4625, Seattle, United States. Association for Computational Linguistics.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 2005. Kea: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pages 129–152. IGI global.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. 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, Online. Association for Computational Linguistics.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):1–37.

Jialin Wu, Jiasen Lu, Ashish Sabharwal, and Roozbeh Mottaghi. 2022. Multi-modal answer validation for knowledge-based VQA. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 2712–2721. AAAI Press.

Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open domain web keyphrase extraction beyond language modeling. 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 5175–5184, Hong Kong, China. Association for Computational Linguistics.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the*

2018 Conference on Empirical Methods in Natural Language Processing, pages 1980–1990, Brussels, Belgium. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Qi Yang, Weinan Wang, Lucas Pierce, Rajan Vaish, Xiaolin Shi, and Neil Shah. 2021. Online communication shifts in the midst of the covid-19 pandemic: A case study on snapchat. *Proceedings of the International AAAI Conference on Web and Social Media*, 15(1):830–840.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. 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 3687–3692, Hong Kong, China. Association for Computational Linguistics.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Struc-

tured relation discovery using generative models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Li Zhang, Shuo Zhang, and Krisztian Balog. 2019a. Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1029–1032.

Shiyue Zhang and Mohit Bansal. 2019. Addressing semantic drift in question generation for semi-supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2495–2509, Hong Kong, China. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura,

Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019b. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. 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 563–578, Hong Kong, China. Association for Computational Linguistics.

Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. 2020. Evaluating commonsense in pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9733–9740.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington, USA. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks*, pages 45–50, Valetta, Malta.