Distilling Word Vectors from Contextualised Language Models

A thesis submitted in partial fulfilment

of the requirement for the degree of Doctor of Philosophy

Yixiao Wang

August 2023

Cardiff University
School of Computer Science & Informatics

Abstract

Although contextualised language models (CLMs) have reduced the need for word embedding in various NLP tasks, static representations of word meaning remain crucial in tasks where words have to be encoded without context. Such tasks arise in domains such as information retrieval. Compared to learning static word embeddings from scratch, distilling such representations from CLMs has advantages in downstream tasks [68],[2]. Usually, the embedding of a word w is distilled by feeding random sentences that mention w to a CLM and extracting the parameters. In this research, we assume distilling word embeddings from CLMs can be improved by feeding more informative mentions to a CLM. Therefore, as a first contribution in this thesis, we proposed a strategy for sentence selection by using a topic model.

Since distilling high-quality word embeddings from CLMs requires many mentions for each word, we investigate whether we can obtain decent word embeddings by using a few but carefully selected mentions of each word. As our second contribution, we explored a range of sentence selection strategies and tested their generated word embeddings on various evaluation tasks. We found that 20 informative sentences per word are sufficient to obtain competitive word embeddings, especially when the sentences are selected by our proposed strategies.

Besides improving the sentence selection strategy, as our third contribution, we also studied other strategies for obtaining word embeddings. We found that SBERT embeddings capture an aspect of word meaning that is highly complementary to the mention embeddings we previously focused on. Therefore, we proposed combining the vectors generated from these two methods through a contrastive learning model. The results confirm that combining these vectors leads to more informative word embeddings.

In conclusion, this thesis shows that better static word embeddings can be efficiently distilled from CLMs by strategically selecting sentences and combining complementary methods.

Acknowledgements

First, I sincerely thank my main supervisor, Professor Steven Schockaert. I am grateful that he accepted me as his PhD student initially. As a student from a philosophical background converting to NLP with purely intellectual interests, my theoretical knowledge, programming skills, and research ability were lacking. Professor Steven Schockaert showed great patience and guided me step by step to build up my profile as an independent NLP researcher. His constant support, guidance, and encouragement over the past four years make my PhD course a great journey.

Second, I would like to thank my second supervisor, Dr Luis Espinosa Anke, for his constructive feedback and thoughtful suggestions that have improved the quality of my research. I also thank Dr Zied Bouraoui for helping me run all the NLP experiments. Moreover, I want to express my sincere gratitude to the School of Computer Science and Informatics, Cardiff University, for providing me with a grand PhD scholarship. The comfortable and lovely environment provided by the school will always be a part of my happy memory.

Finally, I would like to express my deepest gratitude to my wife and parents, whose support, encouragement, and faith in me are always the constant source of my strength through challenges and uncertainties.

Contents

List of Figures

List of Tables

List of Publications

The work introduced in this thesis is based on the following publications:

- Yixiao Wang, Zied Bouraoui, Luis Espinosa Anke, and Steven Schockaert. "Deriving word vectors from contextualized language models using topic-aware mention selection." The 6th Workshop on Representation Learning for NLP (2021).
- Yixiao Wang, Zied Bouraoui, Luis Espinosa-Anke, and Steven Schockaert. "Sentence selection strategies for distilling word embeddings from BERT." 14th Conference on Language Resources and Evaluation (2022)

Chapter 1

Introduction

Natural language processing (NLP), devoted to making a machine understand and respond to human language, is a fast-growing technology that has profoundly shaped the world. Intuitively, we can assume that a machine cannot understand a sentence's meaning without understanding the meaning of its constituent words. Therefore, word representation learning is a fundamental research field of natural language processing. In recent years, the success of pre-trained neural language models has pushed research frontiers profoundly, bringing word representation learning to the next level by making word representations context-sensitive. Motivated by the enormous achievement of pre-trained language models, this thesis tries to exploit such contextual word representations to obtain better word representations. The first part of this chapter introduces the context of this research. Then, the research gaps will be identified in the second part. Subsequently, the research objectives will be clarified, and their significance will be justified. Finally, the limitation of this research will be discussed.

1.1 Background

To learn a language and use it in daily life, people need to start by learning its vocabulary before constructing sentences from a group of words [54]. We assume that this rule also applies to NLP: grasping the meaning of words in a language is the foundation for further downstream NLP tasks. However, teaching a machine to understand the meaning of a word is difficult. This is because words are more than discrete symbols, with each word triggering a mental representation that reveals its meaning and reference [77]. In order to make NLP possible, each word should trigger a machine representation that reflects its meaning. The question is, what kinds of meaning should a representation contain? As [11] suggested, a good representation should express general-purpose priors that are not task-specific but would be likely beneficial for a learning machine to solve AI tasks. In other words, the meaning captured by a word representation should be general and not task-specific; it can be applied to different contexts and provide a solid foundation for different downstream tasks.

However, it is difficult to encode the word meaning into something tangible. Although we can have a discrete symbol representing a concept, its expressed meaning is always out of touch because it is abstract and sealed in the human mind. As [40] points out, a solution might be to represent the meaning of a concept as a distribution of specific properties. For example, the meaning of "*banana*" can be represented by the combination of its properties: being *curved*, *yellow*, and *sweet*. Nevertheless, those defining properties themselves need to be further defined. To avoid this vicious circle, a concept's meaning should be represented by quantitative features that can be measured numerically.

Initially applied in the Information Retrieval (IR) field [91], the vector space model provides an inspiring way to represent meaning. To represent the meaning of documents from a given collection, each document is encoded as a list of features corresponding to all words in the collection. Each feature has a numerical value depending on the number of occurrences of the corresponding word in that document. In other words, the meaning of any document can be described as a distribution among the space composed of word occurrences. Driven by this strategy, [91] obtains vector representations for documents and then confirms the effectiveness of this representation learning by exploring the learned vector space. They found that documents with the same topic tend to cluster together while being apart from the documents in different topics. This research backs up the bag-of-words hypothesis: the word pattern used in a document tends to indicate the document's meaning [46]. Unlike documents, a word is usually an independent lexical unit without an inside feature pattern that illustrate its meaning. Instead, a word's meaning can often be inferred from its surrounding word pattern: context. So, for example, when we read a scientific article containing some words we do not know, we can always infer their meaning from the context where they occur. Firth [36] famously summarised these phenomena as "you shall know a word by the company it keeps." Thus, like documents, word meanings can also be represented by word patterns.

Instead of counting all surrounding words, the word patterns for each word are generally encoded by language models. A language model is a probability distribution model over sequences of words [27, 3]. For example, given an incomplete sentence: "Don't watch this film because it is ( )", a language model should complete the sentence by assigning a higher probability to negative adjective words rather than positive ones. Language models are usually trained on an enormous corpus to predict the following words given an incomplete sentence. Then, given the comparison between its predictions and actual answers, language models can update their hidden parameters to minimise errors and improve their accuracy. By constantly fine-tuning the hidden parameters in the corpus, the language model gradually reaches a stable state that produces correct output given an incomplete sentence. The surrounding word pattern for each word is then encoded by a list of hidden parameters that give the model the best overall performance. In other words, the meaning of a word is represented by the hidden parameters triggered by itself. By extracting the hidden parameters from the well-trained model triggered by each word, all words obtained their unique vector representation. This method of encoding words as fixed-length vectors is also known as word embedding [27, 3].

Intending to obtain the best word embeddings, considerable studies propose diversified frameworks. Among those implementations, Word2Vec [74] and CBOW [80] are the most popular methods of obtaining word embedding due to their efficiency and high quality. In addition, their performance on several evaluation tasks confirms that their word representation preserves word meanings. For example, their vector representations corresponding to similar words are geometrically closer than those less similar in the vector space. [74] reported that the relationship between two words can be defined by subtracting their corresponding word vectors. The following equation can illustrate this inspiring observation:

$$Paris - France = Rome - Italy$$

Furthermore, Word2Vec [74] and CBOW [80] also achieved competitive results on both intrinsic evaluations and extrinsic evaluations, such as word similarity, word analogy, sentiment analysis, and part-of-speech tagging [108].

The word embeddings mentioned above is also regarded as static word embeddings or non-contextual word embeddings. This is because each word type only has a single representation regardless of its context. Although static word embeddings can capture meanings of words at the type level, they struggle with polysemous disambiguation [81, 7]. For example, a static word representation of the English word *"apple"* fails to tell machines whether a high-tech company or a kind of fruit is referred to. CLMs(Contextual language models) solve this question by giving a word different representations. Rather than learning a word embedding for each word at the type level, CLMs, such as ELMo [81] and BERT [29], can learn contextual word representations that vary their value from context to context. For example, their vector representations for the word $apple$ in the context of *"apple announced the fifth generation iPod Touch"* is entirely different from their representations in the context of *"apple trees are cultivated worldwide"*.

Furthermore, as confirmed by several works [100, 32, 25], CLMs have much deeper and more sophisticated neural network structures thus can capture context in more effective way. As the result, CLMs obtain great results on many extrinsic evaluation tasks and outperform NLP models based on static word embedding by a large margin [81, 29]. As a representative CLM, BERT obtained state-of-the-art results on eleven important NLP benchmarks at the time of its introduction. Motivated by this inspiring achievement, many studies have been conducted to probe the knowledge captured by CLMs such as BERT. These works [25, 48, 41, 33] confirm that contextual embeddings from BERT capture rich linguistic knowledge such as syntactic structure, subject-verb agreement, and semantic role. Furthermore, other works [82, 28, 119] reveal that BERT also possesses world knowledge, such as knowing Dante was born in Florence and predicting properties that are never or rarely stated explicitly.

1.2 Research Questions and The Hypothesis

Given the impressive performance of CLMs, static word embeddings have been replaced by CLMs in many NLP applications in the last few years. However, contextual word representations can only represent the word's meaning at the token level because they are overly dependent on their local context. Ethayarajh [32] found that the variation of contextual representation of a single word in BERT is driven by its varying context rather than the word's inherent polysemy. Specifically, they find English stop words, such as "the," "of," and "to," have the most context-specific representations in BERT, even though those stop words have relatively stable meanings in the English language.

Moreover, the representation of the general meaning of an independent word or a concept, regardless of its context, is still required in many NLP applications where word meaning has to be modelled without sentence context. Instances of such applications include ontology alignment [59], ontology completion [65], zero-shot learning [94] and few-shot learning [117, 64, 118]. The ontology alignment task requires the ontologies designed by different experts to be aligned for integrating knowledge. However, different experts use various terms (*"Conference Dinner"* or *"Banquet"*) to denote the same entities. Static word embeddings can be applied to this task by calculating the similarity scores of pairs of terms and determining correspondences between terms. Besides, in [68], they found that static word vectors can be helpful as "anchors" for improving the representations of CLMs. At the same time, [2] also obtained improved results by combining BERT with static word vectors.

Instead of going back to learning static word embedding from scratch, distilling static word vectors from CLMs might be a better way to obtain word representations at the type level. The most apparent motivation is that the rich knowledge preserved in those large CLMs provides an excellent resource for extracting meaningful word representation. The works [25, 48, 119, 82] probing knowledge learned by BERT lend support to the idea that static word vectors induced from CLM have some inherent advantages compared to those from standard word embedding models. The other motivation for distilling static word vectors from CLMs is that we can always build desired word representations by leveraging the context sentences from domains of interest. For example, we can restrict the context of a word to a medicine corpus to obtain the meaning of that word used in the domain of medicine.

Driven by these motivations, some recent works [31, 16, 105] have explored how to distil a static word representation from CLMs. The strategy applied in these works is straightforward. Given a word w, the process of obtaining its static representation from a CLM consist of the following steps:

1. Finding sentences from a corpus that mentions w
2. Feeding sentences as input into a CLM and obtaining contextual representations as output.
3. Obtaining the static representation by aggregating all the contextual representations.

Since usually it is not computationally feasible to run the CLM on all the sentences mentioning w, a sample of such sentences has to be selected. In the works mentioned above, sentences are selected randomly, but random selection may not be optimal. If we want to use the resulting word vectors in downstream tasks such as zero-shot learning or ontology completion, we need vectors that capture the salient semantic properties of words. Intuitively, we should thus favour sentences that best reflect these properties. For instance, many of the mentions of the word *banana* on Wikipedia are about the cultivation and export of bananas, or about some particular banana cultivars. By learning a static word vector from such sentences, we may end up with a vector that does not reflect our commonsense understanding of bananas, e.g. the fact that they are curved, yellow and sweet. This leads to the first research question in this dissertation: how should these sentences be chosen to distil better static word embedding?

Furthermore, a large number of sentences were used to obtain each word embedding in the aforementioned works. This process could be computationally too expensive for many applications, especially when a large vocabulary size is involved. If high-quality static word vectors can be learned from a few informative mentions, the efficiency of distilling word embedding from CLMs would be significantly boosted. Given the need for more research on this specific topic, the second research question in this dissertation is whether high-quality word embeddings can be learned from just a few mentions of each word.

Moreover, different strategies of obtaining word embedding from CLMs may be complementary because they usually capture different aspects of contextual information. For example, from the probing experiments, the representations of the same word from BERT[29] and SBERT[89] tend to capture complementary aspects of its semantic meaning. This motivation leads to my dissertation's third research question: whether combining representations obtained by different methods result in a better static word embedding.

To sum up, the hypothesis in this dissertation is that better static word embeddings can be efficiently distilled from CLMs by strategically selecting sentences and combining the complementary methods. In order to verify this hypothesis, the following research questions are addressed:

1. Can higher-quality word embeddings be obtained by selecting sentences strategically?
2. Can higher-quality word embeddings be obtained from a few mentions of each word?
3. Can higher-quality word embeddings be obtained by combining representations that capture complementary aspects of word meaning?

1.3 Contribution

The primary aim of this thesis is to distil high-quality static English word embeddings from CLMs. The contributions made through this thesis are:

- We have proposed the use of topic models to improve how mentions for each word are sampled. Rather than learning a single vector representation for the target word, we learn one vector for each sufficiently relevant topic. Also, we proposed to construct the final representation of a word w as a weighted average of those different topic-specific vectors. The experiments show that sampling mentions with the topic model can capture a more diversified context, improving the quality of static word embedding distilled from CLMs.
- We have considered the new challenge of distilling high-quality static word embeddings from language models using only a small number of mentions of each word. We propose a range of strategies for sampling mentions. Based on the analysis, we found that using Pointwise Mutual Information (PMI) and definition sentences lead to better static word embeddings. Furthermore, this research confirms that high-quality word embedding can indeed be obtained from just a few highly-informative mentions of each word.

- We have proposed to use SBERT embeddings to capture another aspect of word meaning that is complementary to the word meaning captured by BERT. Then, we found that combining SBERT embeddings and BERT is beneficial in word similarity task and lexical classification task. We also have proposed a strategy for combining word embeddings through a contrastive learning model and distil low-dimensional embeddings. These embeddings have better neighbourhood structure, which result in a impressive result on word similarity task.

The overall experimental results of this research support our research hypothesis: better static word embeddings can be efficiently distilled from CLMs by strategically selecting sentences and combining the complementary methods

1.4 Thesis Structure

The remainder of this thesis is organised as follows:

- Chapter 2 – Background and Related Work – provides an in-depth review of the literature on standard word embeddings, CLMs, ways to distil static word embedding from CLMs, application of word embedding, and word embedding evaluation.

- Chapter 3 – Topic-Aware Mention Selection Strategy – aims to answer the first research question. This chapter proposes the use of topic models to improve how word mentions are sampled. The topic model used in this research is Latent Dirichlet Allocation [14]. First, given the collected contexts of each word, Latent Dirichlet Allocation is applied to partition all the mentions into several clusters corresponding to different topics. Then, topic-specific vectors for each word are obtained and evaluated on several benchmarks to test the effect of our proposed strategy.

- Chapter 4 – Exploring Other Mention Selection Strategies – aims to answer the second question by empirically analyzing a range of strategies for selecting mentions of a given word w. Each strategy selects a few mentions aiming to maximise the captured semantic properties of w. These strategies result in a list of representations for the word w. Then, these word representations corresponding to different strategies are evaluated on the benchmarks.

- Chapter 5 – Combining Complementary Aspects of Word Meaning – is focused on solving the third research question. This chapter proposes combining word representations from different strategies through a contrastive learning model. The masked vector representations from BERT and sentence representations from SBERT are found to be able to capture different aspects of semantic meaning. Given a word w, its mention representation from SBERT and its masked contextual word representations from BERT are extracted before feeding into a proposed contrastive learning model. The contrastive learning model then outputs a static representation for word s which is evaluated on several benchmark to answer the third research question.

Chapter 2

Background and Related Work

2.1 Introduction

This chapter presents a literature review in the area of word embedding, pre-trained language models and contextualized word embedding before discussing the evaluation and application of word embedding. This chapter consists of four sections. Section 2.2 gives a general overview of the motivation for word embedding technology. As the most representative word embedding learning method, Word2Vec is explained in detail to illustrate the essence of word embedding learning. Several important works on developing word embedding are also covered and discussed in this section. As pre-trained neural language models replaced standard word embedding in recent years, this shift caused a huge change in the research map of natural language processing. The motivation for this paradigm shift is explained in Section 2.3, where several pre-trained neural language models are reviewed. As the most representative language model, BERT, with its underlying framework, is explained in detail. The prevalence of pre-trained neural language models has motivate research in probing the knowledge captured by them. Their probing results are also presented and discussed in this section. The positive results from the above probing works caused a booming research topic, namely extracting contextualized word embedding from pre-trained language models. The review of this line of work is discussed in the Section 2.4. The application and evaluation of word embedding is discussed in the Section 2.5

2.2 Word embedding

2.2.1 Motivation of Word Embedding

Daily usage words are more than discrete symbols, with each word having complex meanings that need to be correctly decoded by the human mind. To make a machine interpret and understand words, the meanings of words need to be captured in their representation to facilitate downstream NLP tasks. The question is, what form should word representations take to fulfill this purpose? As [11] suggested, a good representation should express general-purpose priors that are not task-specific but would be likely beneficial for a learning machine to solve AI tasks. In natural language processing, a good representation should capture semantic and common sense knowledge from text data. To encode diverse and complex meanings, the representation of words needs to have enough expressive power to fit language knowledge into a manageable system in which semantic meaning can be measured from the inside.

Initially applied in the Information Retrieval (IR) field, the vector space model has shed light on how to represent semantic information [3]. To build a representation for each document from a collection, [91] encode each document to a real-valued and fixed dimensional vector in which each dimension corresponds to a specific term in the vocabulary list. Therefore, all documents can be described as distributions among the space composed of feature terms. Thus, all terms in the collection formulate a vector space that can describe any document as a vector. This way of obtaining representation using a vector space is usually called distributed representation. It decomposes a subject into a list of feature values, with each element measured by its magnitude on the corresponding feature. Since all documents are represented in a unified vector space, the meanings of documents can be compared with each other. [91] found that semantically similar documents are closer to each other when their euclidean distance is measured. The research from [91] supports the bag-of-words hypothesis: the word pattern used in a document tends to indicate the document's relevance. Put another way, a document's word pattern can capture that document's meaning and be used as a representation.

Unlike document representation, a word is usually an inseparable lexical unit, so it does not have a word pattern inside. Instead, a word's meaning can often be inferred from its surrounding words: context. For example, when we read a scientific article containing some words we do not know, we can always infer their meaning from the context where they occur. Therefore, contextual words can represent the meaning of a target word.

Instead of representing a word by counting its surrounding words, its context can be encoded more effectively by a language model's parameters that assign maximum likelihood to the target word. The fixed dimensional word representations obtained by trained language models are also called word embeddings. The research from [9] confirms that word embeddings significantly outperforms the word vector obtained by counting surrounding words. [10] proposed the first neural network language model that is trained to predict the next word given a preceding context. Neural networks have rich and deep representation power, which can perform many tasks by tuning their parameters from hidden layers. In order to become a language model, a neural network is usually trained on a big corpus, and it constantly predicts the next word via maximum likelihood estimation given a preceding context. Meanwhile, the parameters inside the neural network are constantly tuned based on the feedback after each prediction. Once the neural network reaches the best result in predicting the following words, its well-tuned parameters are frozen and can be taken out as word embeddings to represent the meanings of words.

2.2.2 Word2Vec

Word2Vec [74] is probably the most influential and representative method for learning pre-trained word embeddings. Using an simple and shallow neural network, Word2Vec

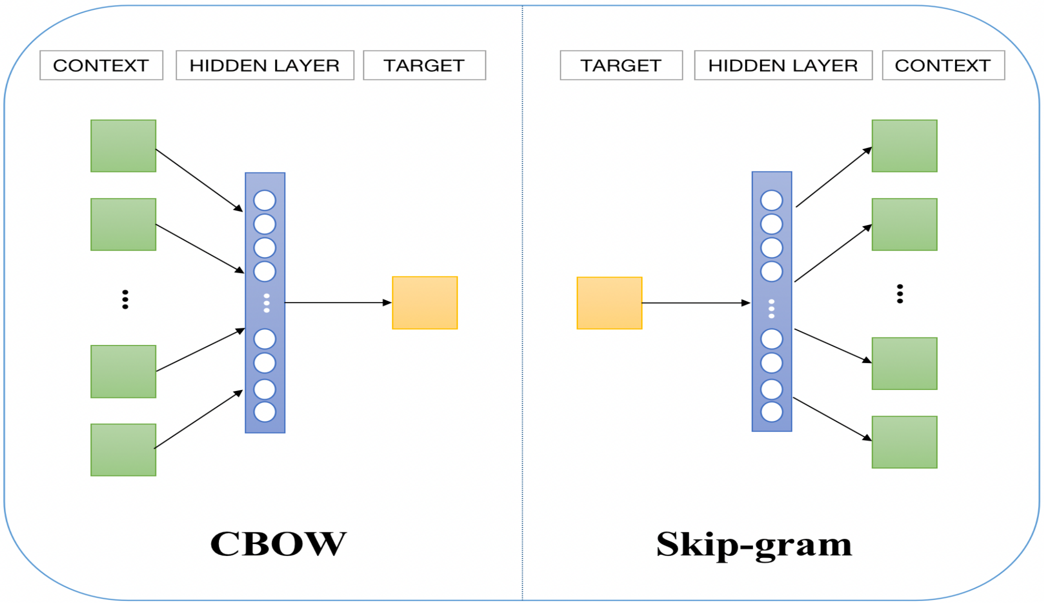

Figure 2.1: A diagram of CBOW and Skip-gram.

can learn representations of all words by pushing each target word closer to its contextual words in the vector space. Word2Vec provides two ways of learning word embeddings, they are Skip-gram and Continuous bag-of-words (CBOW). Both methods are inspired by neural language models while differing in their associated training tasks. A diagram of both CBOW and Skipgram is presented in Figure 2.1.

Skip-gram is trained to predict the N surrounding context words within a fix-sized window from a given target word. The input to the Skip-gram is a one-hot vector of the target word, which is forwarded to a hidden layer of size $V \times d$, where V is the vocabulary size and d is the embedding dimension. The hidden layer output is then passed to a softmax layer, generating N probability distributions over the vocabulary and predicting the most likely context words. Afterward, prediction loss is calculated by comparing the predicted and actual context words. CBOW, in contrast, is trained to predict target words from the surrounding context words. For each word, a fix-sized window of words surrounding the target word is used as the input and fed to the model. As a multi-hot vector, the input passes from the embedding layer to the softmax layer to generate a probability distribution over the vocabulary and predict the most likely word. Like Skip-gram, prediction loss is calculated at the endpoint by comparing the predicted and actual context words.

The objective of both Skip-gram and CBOW is to learn the parameters on the embedding layer by minimizing the prediction loss. Thus, both algorithms are trained in a large corpus while the weights on their embedding layer are constantly updated until the total loss minimization. Meanwhile, the updated weights on the embedding layer push together the representation of the target word and its contextual word while drawing apart the representations of irrelevant words. Once a model is fully-trained on the training corpus, the weights on the embedding layer are frozen. Then, the vector representations of every word in the vocabulary can be directly obtained by taking the learned weights from the embedding layer, either trained by CBOW or skip-gram.

The previous studies confirm the efficiency and effectiveness of word embedding learning methods proposed by Word2Vec . [74] found that the word embeddings learned by CBOW and skip-gram can preserve some semantic and syntactic knowledge. [112, 114] found that the word embeddings learned from Word2Vec poses a solid baseline in several word evaluation benchmarks such as BabelNet Domain and McRae Feature Norms. Besides, these pre-trained word vectors can be further processed to do downstream tasks such as text classification [96, 43], or natural language inference [109, 42].

2.2.3 Development of Word Embeddings

Word2Vec is criticized as only focusing on local word co-occurrence while ignoring the global statistics: the learned representation of a word is only affected by the surrounding words. So, for example, a stop word "the" might always co-occur with the target word, say apple, but the word "the" does not tell us anything informative about what an apple is. Glove[80], as the other most influential word embedding learning method, can solve this question. By building a global co-occurrence matrix (word by context) from the corpus, the frequency of words co-occurrence with a given context becomes visible. Then, word embeddings can be obtained by factorizing this huge co-occurrence matrix into a low-dimensional matrix.

The above mentioned two word embeddings learning methods takes the context of a word to be words that precede and follow the target word. However, words that fall in this window might be irrelevant to the target word. For example, in the sentence "Australian scientist discovers star with telescope, word "Australian" modifying the word "scientists" is not informative to reveal the meaning of the word "discovers." The above issue can be solved if context can be derived based on the target word's syntactic relation. [63] obtain the syntactic context of a word by extracting dependency parse trees from their original context. With the help of parsing technology, the syntactic dependency trees can be directly extracted with decent accuracy. Afterward, dependency trees can identify the syntactic relations between contextual words and the target word. Therefore, only the words that have syntactic relation with the target word are considered context. Compared with Word2Vec, this change of context improves the performance of the resulting word embeddings on several evaluation tasks, revealing its better capability of capturing syntactic similarity.

A problem that the early word embedding learning technologies suffer from is the out-of-vocabulary issue [60]. Sometimes, words that are not in the training vocabulary might be found in new data. For example, the word "TensorFlow" might be spotted in AI articles by a model that does not have this word in its training vocabulary even though it contains "tensor" and "flow" as two recognized words. This problem can be eased if the embedding learning model can utilize the internal structure of the words. FastText [15] propose an extension of the skip-gram model to use the internal structure of a word to improve word vector representations. Instead of learning vectors for words directly, FastText represents each word as an n-gram of characters. This change helps to capture the inner structure of words and forces the embedding model to understand the meaning of suffixes and prefixes. Once the word has been represented as character n-grams, a skip-gram model is trained to learn the embeddings. Since the model explores the inner structure of words, FastText can learn representations for the out-of-vocabulary words.

All the embedding learning technologies described above are trained on an unstructured corpus with free text data. In fact, the knowledge about a target word revealed by its context in free text is quite limited. For example, the semantic meanings of a word are unlikely to be explicitly stated in a random sentence that mentions it. That is because the daily use of language is more likely to use words as tools for communication rather than giving full explanations. Therefore, utilizing structured datasets such as knowledge graphs to further enhance the word representation is a natural strategy. To further enrich the word representation power, ConceptNet-Numberbatch [95] utilizes the encoded knowledge in ConceptNet, a semantic network of knowledge about word meanings. First, they convert the pruned ConceptNet to a symmetric term-term matrix in which each cell represents the sum of all edges through which two terms are related in some way. Then, the context for each word in this matrix is defined as the terms that have a direct edge connecting them. Truncated SVD is further applied to this matrix to reduce the number of its columns to 300. Afterward, they apply retrofitting technology to push representations of each word towards its original representation and the representations of its context words. Since retrofitting can enhance any pre-trained embeddings with a knowledge graph, it paves a way to combine the strength of word embeddings learned from unstructured and structured data.

Learning word embeddings in multilingual vector spaces is also confirmed to be helpful in improving monolingual word embeddings. For example, given the embedding spaces of two different languages and a dictionary mapping their vocabularies in translation, [34] project them onto a common vector space such that their translated pairs are maximally correlated. Specifically, this maximal correlation is realized by applying canonical correlation analysis (CCA), which measures the linear relationship between two vectors and finds two optimal projection vectors that maximize the correlation between their transformed vectors. They report that the transformed word embeddings in the common vectors space outperforms its original word embeddings on lexical semantic evaluation tasks.

2.3 Contextualized Language Models

The word embedding technology mentioned above is widely regarded as static word embeddings or non-contextual word embeddings. This is because each word type only has an independent global representation regardless of its context. Although static word embeddings can capture semantic meanings of words at the type level, they struggle with polysemous disambiguation because of their context-free nature. For example, a static word representation of the English word "apple" fails to tell readers whether a high-tech company or a kind of fruit is referred to. Moreover, words in language usage develop their semantical meaning based on the previous words. For instance, an adjective word before the noun word might modify the meaning of the noun word significantly. To solve these issues, pre-trained language models take on this challenge and explore a new strategy for word representation learning. Rather than learning a word embedding for each word at the type level, pre-trained language models such as ELMo and BERT are committed to learning contextual word representations that vary their value from context to context. This idea is realized by representing each word as a function of the entire input sequence that mentions the word.

2.3.1 The Early Pre-trained Language Models

ELMo [81] is one of the earliest works on pre-trained language models as presented in Figure 2.2. It adopts L-layers of bidirectional LSTM (Long short-term memory) [50] (a forward LSTM and a backward LSTM applied to encode both left-to-right and right-to-life contexts) as feature extractor, and it is trained on a large corpus. The training objective is the maximum likelihood of the language models in both directions. Driven

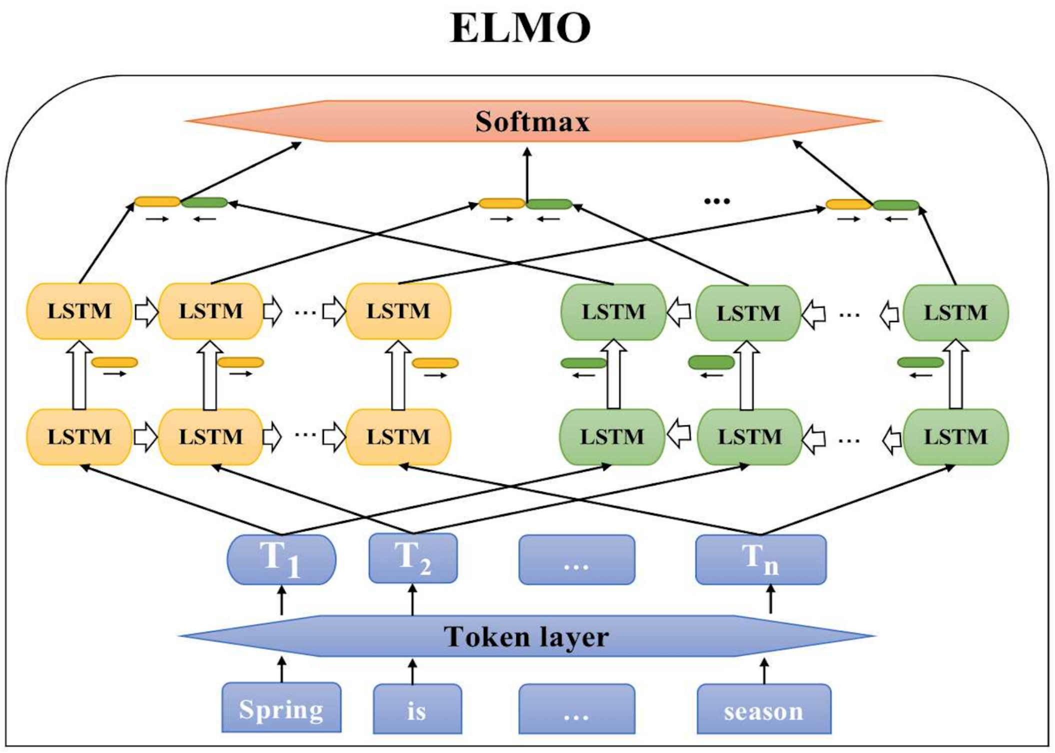

Figure 2.2: A diagram of ELMO.

by this training objective, this bidirectional LSTM-based model needs to correctly predict the following word based on the preceding words in a sentence and predict the preceding word based on the following words. Given a sentence, ELMo splits the sentence into a list of words and represents each word as convolutions of characters before mapping them to context-independent token representations. By splitting the input word into groups of characters, ELMo follows the spirit of FastText to tackle the out-of-vocabulary problem. Then, these context-independent token vectors are fed into a Bidirectional LSTM in which two hidden representations are assigned to each token. Since LSTM is a sequential feature extractor which can store preceding information, the contextual information of the input sentence is retained at hidden parameters corresponding to each token. Therefore, representing each token as the hidden parameters from an LSTM allows a word representation to be contextual. Finally, by aggregating the contextual representations on all bidirectional LSTMs and the non-contextual representation from the corresponding position, each word obtains a single vector as its contextual representation in that specific sentence. The power of contextual word representation from ELMo is proved by its state-of-the-art performance across six challenging NLP problems including question answering, textual entailment and sentiment analysis, according to [81].

Although bi-directional LSTMs can encode the context from both directions, it is not designed to encode the interaction between the contextual words from two sides of the target word due to its sequential natural. Furthermore, LSTMs as a feature extractor has several limitations that make it less effective compared with Transformer, a counterpart that is released later. First, LSTMs is a sequential model that processes sentence word by word so that it does not allow parallel computation. Moreover, the long-term dependency problem often affects all the RNN-based models [84, 56].To encode context better and more efficiently, Transformer [104] replace the LSTMs and almost become the universal feature encoder in NLP applications. On the one hand, Transformers can directly capture the dependency between every two tokens in the given sequence, which overcome the issue of long-range dependency in language. On the other hand, Transformer architecture is not a sequence model; therefore, it allows parallel computation and can further boost training efficiency. Therefore, most of the later released pre-trained language models [29] [85] [70] use Transformer architecture as the encoder for context information. OpenAI-GPT [85] is the first pre-trained language model that uses Transformers as the context feature encoder and the decoder. However, instead of using a bidirectional language model as the training target, GPT's training target is a forward language model which only uses the context preceding the target word. This training objective encourages the model to learn language generation and thus comes with the cost of losing contextual information after the target word.

2.3.2 BERT and Its Extensions

BERT(Bidirectional Encoder Representations from Transformers) [29] is the paradigm-shifting research that dramatically changed the research map in NLP. Like ELMo and

OpenAI-GPT, BERT is also committed to learning contextualized word embeddings through a pre-trained neural network language model. However, BERT takes a different approach from ELMo and GPT regarding the training objective. The forward language model used in GPT ignores the context after the target word that might contain relevant information about the semantic meaning of the target word. Although the bidirectional language model used in ELMo can capture the context from both sides, it still cannot encode the interaction between the left and right contexts. For many NLP downstream tasks, it is vital to use all context information simultaneously. To fully explore the contextual information of a target word, [29] propose a masked language model as the new training objective.

Instead of predicting the target word from either the left or right direction, a masked language model masks a target word and then predicts what that masked word is based on all surrounding words in the sentence. For example, a target word "apple" is masked in the following sentence: "In Autumn the [MASK] fall from the tree.", and the model needs to predict what [MASK] is by inferring from its left context: "in Autumn" and its right context: "fall from the tree." Therefore, a masked language model can encode contexts in both directions simultaneously to better represent the meaning of a target word. Besides the masked language model, the other training task for BERT is next sentence prediction: BERT is also trained to predict whether two given sentences have a sequential connection. This training task encourages BERT to capture the semantic meaning at the sentence level.

Since knowledge about the framework of BERT is essential to understand this thesis, the structure of BERT is explained here. First, before feeding into the model, the input sequence of words from the training corpus is preprocessed by tokenizing them into a list of tokens. By doing so, words are split into smaller subwords so that BERT represents a word as a collection of characters. Meanwhile, each word in the input sequence has a 15 percent chance of being replaced with a [MASK] token. Given a mask token [MASK] and other corrupted word tokens, the model is trained to predict

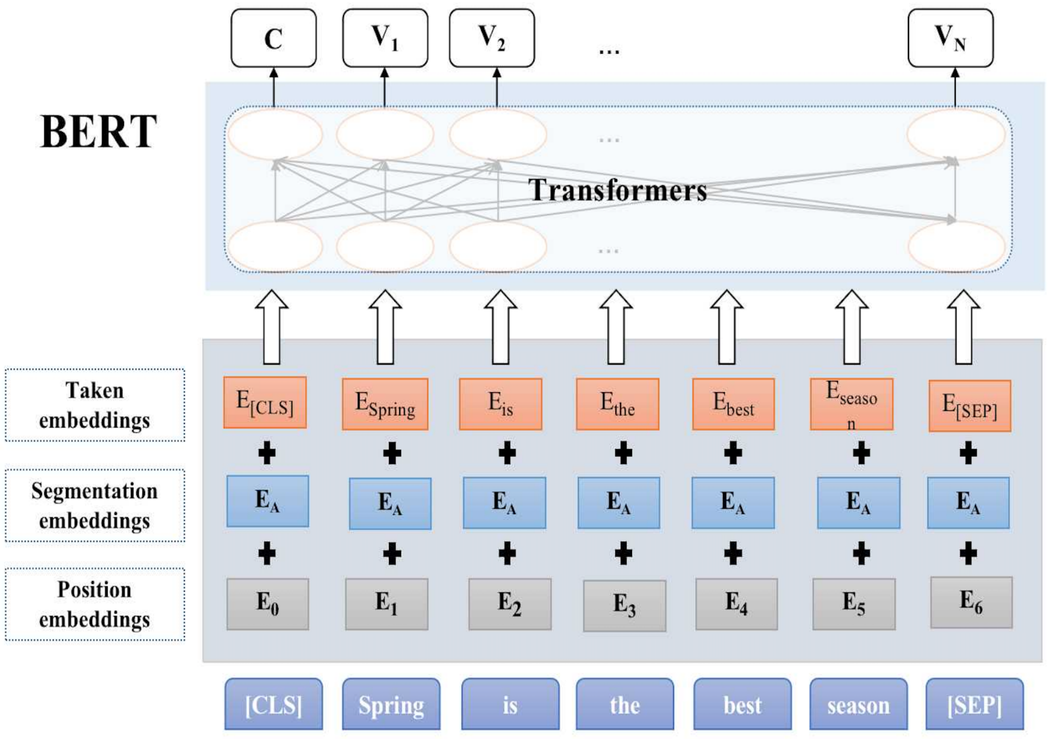

Figure 2.3: A diagram of BERT.

its original word based on its context. In addition, a classification token [CLS] is added at the beginning of the token list to encode the sentence's meaning, and the sentence-ending token [SEP] is added at the corresponding position in the token list to mark the end of a sentence.

Given a list of input tokens, BERT uses three separate embeddings for each token. The diagram of BERT as presented in 2.3 illustrate its workflow.

- **Token embeddings**: static trainable vector representation of each input tokens.
- **Position embeddings**: pre-defined vector indicating the token's position which can preserve the sequential order of words in a sentence.
- **Segmentation embeddings**: used to distinguish two input sentences in support of next sentence prediction task.

Then, these three input embeddings are fed to a stack of bidirectional transformer encoders from bottom to top. The output of each bidirectional transformer encoder is a sequence of vectors, in which each vector corresponds to an input token with the same index. Each bidirectional transformer encoder extracts features from the vectors it received from the preceding level before feeding the generated sequence of vectors to the transformer encoder at the next level. At the top of the final transformer encoder, the output vectors are multiplied by a learned set of classification weights before passing them onto a softmax layer. In the end, the probability distribution over the vocabulary for each masked token is computed to yield the prediction. The cross-entropy loss from these predictions drives the training process for all the parameters in the model.

Once training is completed, the parameters from the token embedding layer and all transformer encoders are frozen. Given a sequence of words as input, the token embedding layer of BERT generates a static vector for each token that remains stable regardless of its context. On the other hand, each transformer encoder generates a contextualized vector for each token, which reflects the meaning of the token in the context. If a word does not have a corresponding token in the vocabulary, its vector representation can be obtained by aggregating the representations of its sub-token.

Liu et al [70] suggests that BERT is significantly under-trained, and they proposed RoBERTa, an optimized version of BERT, by modifying the key hyperparameters in BERT. For example, they use much larger mini-batches and learning rates to train the model. Also, they remove BERT's next-sentence pre-training objective because this change slightly improves the model's downstream task performance. Those modifications of hyperparameters lead RoBERTa to state-of-the-art results on several downstream tasks including GLUE (General Language Understanding Evaluation) [107], SQuAD (The Stanford Question Answering Dataset) [87], and RACE (ReAding Comprehension from Examinations) [62]

Zhang et al [121] propose a knowledge-enriched language model ERNIE, a knowledge-enriched version of BERT. ERNIE is pre-trained on both corpus and knowledge graph.

Furthermore, by encoding knowledge graph structure into knowledge embeddings and aligning them with text, ERNIE integrates entity representations in the knowledge graph into the language model. Therefore, ERNIE not only inherits the linguistic knowledge from BERT by pre-training the model on a large corpus but also obtains knowledge of the world through their designed training objective. Their experimental result shows that ERNIE outperforms BERT on entity typing and relation classification tasks and is comparable with BERT on GLUE tasks.

2.3.3 Probing The Knowledge Learned by BERT

Since BERT obtained state-of-the-art results on eleven natural language processing benchmarks, many researchers are inspired to probe what knowledge is captured by BERT. By analyzing the attention mechanisms of pre-trained language models, [25] found that BERT can capture the syntactic structure of a free text. [48] found that Syntax trees are embedded in a linear transformation of a BERT's word representation space. Moreover, knowledge of subject-verb agreement [41] and semantic roles [33] are also reported as existing in BERT. One line of work has focused on analyzing to what extent language models understand the properties of simple concepts [37] [115] [67]. Their finding shows that language models outperform word embedding models such as Word2Vec and GloVe.

Besides linguistics, various forms of world knowledge are also captured in BERT. For example, [82] find that BERT, without fine-tuning, can correctly predict masked objects given a cloze sentence such as "Dante was born in [MASK]." Similarly, without fine-tuning, [28] shows that pre-trained BERT can do novel common-sense knowledge mining by determining the validity of a given tuple. Finally, [119] found that BERT can achieve state-of-the-art performance on several knowledge graph completion tasks by fine-tuning the pre-trained language model on a knowledge graph completion task. These findings suggest that BERT obtained knowledge about the world during pre-training.

However, a survey conducted by [99] demonstrates that although BERT can capture syntactic structure well, it does not improve performance on tasks that require more semantic understanding. Furthermore, some research [37] [83] reveals that BERT cannot do common-sense reasoning unless the knowledge is explicitly written down. In contrast, [93] found that language models can predict some properties that are never or rarely stated explicitly. However, the model suffers from over-generalization.

2.4 Obtaining Static Word Embedding from BERT

The rich knowledge stored in the BERT suggests that its contextualized vectors can represent the meaning of words. However, unlike standard word embeddings, BERT is designed to solve the NLP tasks directly. Instead of generating pre-trained word features and applying them to different frameworks for downstream NLP tasks, the BERT provides a complete pipeline from generating pre-trained word vectors to deploying them to various downstream NLP tasks. Therefore, contextualized word vectors from BERT are regarded as intermediate results that can only represent the word's meaning at the token level: the representation constrained by its local context. [32] finds that a word's varying contexts, rather than its inherent polysemy, drives its representations to vary in BERT. For example, they find English stop words, such as "the," "of," and "to," have the most context-specific representations in BERT, even though those stop words have relatively stable functions in the English language. Therefore, the representation of the general meaning of a word, regardless of its context, cannot be replaced by its contextualized representation.

Rather than going back to standard word embeddings, the works mentioned above support the idea that word vectors induced from language models may have some inherent advantages. As the analysis from [67] revealed, different word embeddings often have complementary strengths. In particular, [30] argued that learning vectors for a large vocabulary remain an important advantage of static word embeddings. Therefore, obtaining static word embeddings from pre-trained language models receives increased attention in the NLP community. The easiest way to get static word representation from a pre-trained contextual language model is by feeding the model a single word and extracting the output vector as the representation of that word. However, the word representation generated in this way is far from ideal [17]. [32] was one of the first to distill static word vectors from language models as a mechanism for probing how models such as BERT and GPT-2 [86] capture word meaning. They apply PCA (principle component analysis) on contextualized vectors from sampled sentences mentioning the target word and regard their first principal components as the static word vector for that word. In particular, their static vector extracted from lower layers of BERT outperforms Glove and FastText on several benchmarks.

[17] compared different strategies for pooling contextualized vectors and getting representations of words that are split into multiple sub-word tokens. For each word to be represented, they sample sentences that mention it and obtain a set of contextualized representations of that word by aggregating the token representations of its sub-words. Then, they obtain the static word vector by aggregating these contextualized representations. Among different strategies for pooling contextualized word vectors, they found in their experiments that averaging is the best option. Furthermore, they found that using 500 sentences for each word led to much better representations than 10 or 100 words. Another finding from this paper is that the performance of the word vectors can differ substantially depending on which layer of the language model is used for obtaining them. Specifically, they suggest that the optimal layer depends on the number of sampled mentions, with later layers performing better when many mentions are used.

Building on this insight, [106] found that, on several evaluation tasks, averaging the representations of the bottom k layers can lead to better results than using the vectors from any individual layer. Although the optimal value of k depends on the language, task, and configuration, averaging the representations across all layers consistently provided close-to-optimal results. These results seem to imply that the knowledge learned by BERT is distributed across all hidden layers while word meaning at the type level is stored at the bottom layers.

This observation is slightly different from the finding of [67], who tried to select optimum layers of BERT based on validation data. For the word classification benchmarks, the performance of using the last layer is often similar to either averaging the first k layers or selecting a single layer based on the validation data. In addition, [67] also proposed masking the target word when computing the contextualized vectors. There are two benefits of applying a mask to the target word when extracting its representation from a language model. First of all, masking the target word force the model to infer the semantic properties of the marked word rather than irrelevant properties such as the frequency of input tokens. Second, a static representation of that word can be obtained as a single vector rather than the aggregation of representations of several sub-word tokens. Consequently, they only obtain vectors from the final layer, given that the [MASK] token makes the early layers less informative. For most classification datasets, they found that masking the target word led to better results. However, in word similarity benchmarks, vectors obtained with masking under-performed. This discrepancy in the two evaluation benchmarks reveals that word vectors obtained in different ways have complementary strengths.

Several strategies that do not rely on averaging contextualized vectors have also been proposed. For instance, inspired by GloVe word embedding learning, [38] directly obtain a semantic co-occurrence matrix from BERT and uses it as input to the GloVe word embedding method. The benefit of obtaining a semantic co-occurrence matrix from BERT is that informative word pairs are no longer limited by the local window used in GloVe. Their experiments on several word similarity datasets show that the obtained new static vectors can outperform GloVe. Similarly, [44] use a Word2Vec inspired model, which uses BERT to obtain a vector representation of the context of the target word, either a sentence or a paragraph. Following the training method of CBOW, they train a shallow neural network that pushes together the representations of the target word and the representation of its context in the vector space. However, their method performs better than averaging strategies only if a large number of occurrences of each word were used. A somewhat similar idea was pursued by [113], who proposed a modification of Skip-gram in which BERT encodings were used to represent contexts.

Instead of learning a single static vector for each word, another line of research focuses on extracting multiple representations corresponding to different aspects of each word. For instance, [24] obtain mentions of the same word from BERT and then cluster the contextual word vectors into multi-prototype vectors. Their resulting multi-prototype representation is then used to compute word similarity adaptively. Similarly, [5] cluster contextual word vectors for word sense induction. In addition, [102] showed that clustering the contextual representations of a given set of words can produce clusters of semantically related words, which were found to be similar in spirit to LDA topics.

Besides the aforementioned work focusing on distilling word representations, there is another line of research on deriving sentence embeddings from pre-trained language models. Although sentence representations can be directly obtained from BERT by feeding sentences and extracting [CLS] token embedding, the quality of those sentences representations are found to be worse than averaging GloVe embeddings of words in the input sentences [89]. Almeida and Xexeo [89] present SBERT, (Sentence-BERT), a modification of the BERT network to derive fixed-size sentence representations. By adding a pooling layer and siamese and triplet networks on top of BERT, SBERT is fine-tuned on natural language inference datasets to generate similar representations for semantically similar sentences. SBERT obtained state-of-the-art performance on STS benchmark [22] and SentEval benchmark [26], revealing its effectiveness in capturing sentence meaning and potential to reinforce word embeddings.

2.5 Application and Evaluation of Static Word Embedding

2.5.1 Applications of Static Word Embedding

In many NLP tasks, Static word embedding is no longer needed because of the effectiveness of contextualized language models. However, the representation of the general meaning of an independent word or a concept, regardless of its context, is still required in some NLP applications where word meaning has to be modelled without sentence context. For instance, query terms often need to be modelled without context in information retrieval, and word vectors are normally used for this [79, 52]. Instances of such applications even extend to ontology alignment [59], ontology completion [65], zero-shot learning [94] and few-shot learning [117, 64, 118].

The ontology alignment task aims at integrating knowledge representation models designed by different experts. However, different experts may use various terms (*"Conference Dinner"* or *"Banquet"*) to denote the same entities. Therefore, generating a set of correspondences between the entities of different knowledge representation models is essential to ontology alignment [101]. Static word embeddings can be applied to this task by calculating the similarity scores of pairs of terms and determining correspondences between terms, thus aligning ontologies [120, 59]. In the same spirit, static word embedding can be applied to ontology completion tasks. Ontology completion is finding missing plausible but not logically deducible relations from the given ontology [66]. As a simple example, consider the following rules.

$$Beer(x) \rightarrow R(x)$$

$$Gin(x) \rightarrow R(x)$$

Without knowing what the predicate R represents, we can plausibly infer the following rule:

$$Wine(x) \to R(x)$$

The reason behind this inference is that all natural properties shared in common by *beer*, *gin* should also be satisfied by *wine*. Since these natural properties can be captured by word representations at the type level, we can find all the concepts that have similar vector representations with the given concepts and find plausible rules to complete ontologies.

Static word embeddings are also useful in zero-shot learning and few-shot learning where mapping vector space from one domain to another is needed (e.g. mapping images to labels). Many applications require classifying instances whose classes have not been seen previously [111]. Considering the following example. Suppose the training set of an image classification doesn't have or only have a few instances of the label *"cat"* while having enough instance of the label *"dog"*. Can the classifier recognize the cats shown in the above test set? The answer is YES if the classifier knows that *"dog"* and *"cat"* are similar in vector space of words. Thus, learning a mapping from the vector spaces of images to a vector space of words is beneficial for few-shot or even zero-shot learning, especially if the vector space of words can capture salient semantic properties.

2.5.2 Evaluations of Static Word Embedding

Evaluating vector representations of words is difficult. This is because each entry value of a vector is measured against the features corresponding to context patterns which are hard to interpret. Unlike categories designed by human experts, the model learns these features corresponding to statistic patterns to maximise its overall accuracy in the training task. Although there have been some works [76, 71, 92, 20] on learning interpretable word embedding, most word embeddings are hard to interpret. Therefore, various methods have been proposed to evaluate the quality of a word embedding. Those evaluation methods can be divided into extrinsic and intrinsic evaluations [108]. Intrinsic evaluations test the quality of word embeddings by measuring their syntactic or semantic relationships. In contrast, extrinsic evaluations test the quality of a word embedding by measuring how much improvement it brings to downstream NLP tasks. Word embeddings are supposed to perform well in both extrinsic and intrinsic evaluation.

The most common method of intrinsic evaluations is the Word Analogy task. The purpose of this task is to rank word pairs based on their degree of similarity. The similarity between representations of two words is normally calculated by their cosine similarity defined by:

$$cos(w_i, w_j) = \frac{w_i \cdot w_j}{||w_i|| \cdot ||w_j||}$$

This ranking is compared with a gold standard ranking obtained from human judgement. This task aims to measure how well the human perceived similarity is captured by the word vector representations [108]. However, the similarity of words pair may be confused with their relatedness [35]. For example, *sea* and *lake* are two similar words, while *sea* and *beach* are two related words. Therefore, word similarity tasks are divided into semantic similarity and relatedness tests. The datasets that are normally used for semantic similarity tests are *SimLex999*[49], *SemEval-17*[21], and the semantic similarity portion of *WordSim*[1]. The datasets that are used for the semantic relatedness are *MEN*[18], *MTurk-771*[45], and the semantic relatedness portion of *WordSim*[1].

Another kind of intrinsic evaluation task is t the lexical classification task. This task focuses on predicting the semantic properties of individual word representations. For each semantic property, a separate binary classifier is trained on the training set containing the positive and negative words. The trained classifier is then evaluated based on its predictions on the test set by calculating the F1 score. The classifier's performance on the test set is regarded as an indicator of how well the word embeddings capture semantic properties. According to [67], lexical semantic properties can be further divided into commonsense properties (e.g. falcons have wings), taxonomic properties (e.g. falcons are a type of bird) and topic properties(e.g. falcons are related to the sky). There are two datasets specifically focusing on commonsense properties: the extension of the McRae feature norms dataset [72] that was introduced by Forbes et al. [37][1] and the CSLB Concept Property Norms[2]. WordNet supersenses dataset[3], which groups nouns into broad categories, focuses on taxonomic properties. The dataset normally used for evaluating topics properties is BabelNet domains dataset[4] [19], which assigns lexical entities to topic domains.

Although there are also commonly used extrinsic evaluation tasks such as POS (Part-of-speech) tagging and text classification, ontology completion might be a more relevant extrinsic evaluation task for static word embeddings. This is because the ontology completion task is an application where word meaning has to be modelled without sentence context. In this task, word vectors are input features to a graph neural network whose structure is determined by a given ontology or rule base. In particular, given a rule template such as $\star(x) \wedge \textit{LocatedIn}(x, y) \rightarrow \textit{CapitalCity}(y)$, the task is to predict which concepts can be used for the placeholder $\star$ to make the rule plausible. Four well-known domain-specific ontologies are normally used for ontology completion [66]: Wine[5], Economy[6], Olympics[7] and Transport[8]. Another benchmark for this task is SUMO[9], a large open domain ontology.

[1]https://github.com/mbforbes/physical-commonsense
[2]https://cslb.psychol.cam.ac.uk/propnorms
[3]https://wordnet.princeton.edu/download
[4]http://lcl.uniroma1.it/babeldomains/
[5]https://www.w3.org/TR/2003/PR-owl-guide-20031215/wine
[6]http://reliant.teknowledge.com/DAML/Economy.owl
[7]http://swat.cse.lehigh.edu/resources/onto/olympics.owl
[8]http://reliant.teknowledge.com/DAML/Transportation.owl
[9]http://www.adampease.org/OP/

2.6 Summary

In this chapter, we have discussed the background knowledge related to word embedding learning and contextualized language models. Several important recent works on distilling static word embeddings from contextualized language models are also discussed in this chapter. These works provide valuable insight into effectively distilling static word embeddings from CLMs and pave the foundation for future study. However, there is a noticeable research gap in this line of work. Most of the works mentioned above ignore the association between the quality of mentions of words and the quality of resulting word representations. They feed random sentences that mention the target word as input to CLMs and extract its vector representations. This method is less than optimal because random mentions of a word might not reveal its semantic properties vital to the downstream application discussed earlier. The next three chapters will explain how we fill this research gap and distil word embeddings from contextualized language models using different mention selection strategies.

Chapter 3

Topic-Aware Mention Selection

3.1 Introduction

In order to distil static representation of a word w from CLMs, randomly selecting sentences that mention w may not be optimal. If we want to use static word vectors in downstream tasks such as zero-shot learning or ontology completion, we need vectors that capture the salient semantic properties of words. Intuitively, we should thus favour sentences that best reflect these properties. For instance, many of the mentions of the word *banana* on Wikipedia are about the cultivation and export of bananas, and about the specifics of particular banana cultivars. By learning a static word vector from such sentences, we may end up with a vector that does not reflect our commonsense understanding of bananas, e.g. the fact that they are curved, yellow and sweet.

The main aim of this chapter is to answer the first research question: can higher-quality word embeddings be obtained by selecting sentences strategically? To answer this question, we analyze to what extent topic models such as Latent Dirichlet Allocation [13] can be applied to selecting sentences and improving word representations. Continuing the previous example, we may find that the word *banana* occurs in Wikipedia articles on the following topics: industry, biology, food or popular culture. We assume that the topic of food should be a better topic from which to get mentions of *banana* than topics of industry and biology. This is because they are more likely to reveal bananas' commonsense properties (e.g., edible and sweet). Nevertheless, we found that most of the mentions of *banana* in Wikipedia articles are about the topics of industry and biology. Therefore, we assume that mentions selected based on relevant topics are more informative than random mentions in Wikipedia articles. Note that the optimal selection of topics is task-dependent, e.g. in an NLP system for analyzing financial news, the economics topic would clearly be more relevant. For this reason, we propose to learn a word vector for each topic separately. Since the optimal choice of topics is task-dependent, we then rely on a task-specific supervision signal to make a soft selection of these topic-specific vectors. These topic-specific vectors for each word should capture more salient semantic properties and outperform the vectors obtained from randomly selected mentions in lexical classification tasks.

Another important question is how CLMs should be used to obtain contextualized word vectors. Given a sentence mentioning w, a model such as BERT-base constructs 12 vector representations of w, i.e. one for each layer of the transformer stack. Previous work has suggested to use the average of particular subsets of these vectors. In particular, Vulic et al. [105] found that lexical semantics is most prevalent in the representations from the early layers, and that averaging vectors from the first few layers seems to give good results on many benchmarks. On the other hand, these early layers are least affected by the sentence context [31], hence such strategies might not be suitable for learning topic-specific vectors. [67] explore a different strategy, which is to mask the target word in the given sentence, i.e. to replace the entire word by a single [MASK] token, and to use the vector representation of this token at the final layer. Their resulting vector representations thus specifically encode what the given sentence reveals about the target word, making this a natural strategy for learning topic-specific vectors.

Note that there is a clear relationship between this latter strategy and CBOW [73]: where in CBOW the vector representation of w is obtained by averaging the vector representations of the context words that co-occur with w, we similarly represent words by averaging context representations. The main advantage compared to CBOW thus comes from the higher-quality context encodings obtained using CLMs. The main challenge, as already mentioned, is that considering all the mentions of w and distilling vectors from contextualized language models can be computationally expensive, whereas this is typically feasible for CBOW (and other standard word embedding models). Our contributions in this chapter can be summarized as follows[1]:

- We analyze different strategies for deriving word vectors from CLMs, which rely on sampling mentions of the target word from a text collection.
- We propose the use of topic models to improve how these mentions are sampled. In particular, rather than learning a single vector representation for the target word, we learn one vector for each sufficiently relevant topic.
- We propose to construct the final representation of a word w as a weighted average of different vectors. This allows us to combine multiple vectors without increasing the dimensionality of the final representations. We use this approach for combining different topic-specific vectors and for combining vectors from different transformer layers.

3.2 Constructing Word Vectors

In Section 3.2.1, we first describe different strategies for deriving static word vectors from CLMs. Section 3.2.2 subsequently describes how we choose the most relevant topics for each word, and how we sample topic-specific word mentions. Finally, in Section 3.2.3 we explain how the resulting topic-specific representations are combined to obtain task-specific word vectors.

[1]All code and data to replicate our experiments is available at `https://github.com/Activeyixiao/topic-specific-vector/`.

3.2.1 Obtaining Contextualized Word Vectors

The basics of the BERT contextualised language model was introduced in the Chapter 2. Let us write $\mathbf{w}_i^s$ for the representation of word w in the ith transformer layer. We will refer to the representation in the last layer, i.e. $\mathbf{w}_{12}^s$ for BERT-base and $\mathbf{w}_{24}^s$ for BERT-large, as the output vector. Given a sentence s in which the word w is mentioned, there are several ways in which BERT and related models can be used to obtain a vector representation of w. If w consists of a single word-piece, a natural strategy is to feed the sentence s as input and use the output vector as the representation of w. However, Vulic et al [106] have found that it can be beneficial to also take into account some or all of the earlier transformer layers, where fine-grained word senses are mostly captured in the later layers [88] but word-level lexical semantic features are primarily found in the earlier layers [105]. For this reason, we will also experiment with models in which the vectors $\mathbf{w}_1^s, ..., \mathbf{w}_{12}^s$ (or $\mathbf{w}_1^s, ..., \mathbf{w}_{24}^s$ in the case of BERT-large) are all used. In particular, our model will construct a weighted average of these vectors, where the weights will be learned from training data (see Section 3.2.3). For words that consist of multiple word-pieces, following common practice, we compute the representation of w as the average of its word-piece vectors. For instance, this strategy was found to outperform other aggregation strategies in Bommasani et al. [16].

We will also experiment with a strategy that relies on masking. In this case, the word w is replaced by a single [MASK] token (even if w would normally be tokenized into more than one word-piece). Let us write $\mathbf{m}_w^s$ for the output vector corresponding to this [MASK] token. Since this vector corresponds to BERT's prediction of what word is missing, this vector should intuitively capture the properties of w that are asserted in the given sentence. We can thus expect that these vectors $\mathbf{m}_w^s$ will be more sensitive to how the sentences mentioning w are chosen. Note that in this case, we only use the output layer, as the earlier layers are less contextual [32].

To obtain a static representation of w, we first select a set of sentences $s_1, ..., s_n$ in which w is mentioned. Then we compute vector representations $\mathbf{w}^{s_1}, ..., \mathbf{w}^{s_n}$ of w

from each of these sentences, using any of the aforementioned strategies. Our final representation $\mathbf{w}$ is then obtained by averaging these sentence-specific representations, i.e.:

$$\mathbf{w} = \frac{\sum_{i=1}^{n} \mathbf{w}^{s_i}}{\| \sum_{i=1}^{n} \mathbf{w}^{s_i} \|}$$

3.2.2 Selecting Topic-Specific Mentions

To construct a vector representation of $\mathbf{w}$, we need to select some sentences $s_1, ..., s_n$ mentioning w. While these sentences are normally selected randomly, our hypothesis in this chapter is that purely random strategies may not be optimal. Intuitively, this is because the contexts in which the target word w is most frequently mentioned might not be the most informative ones, i.e. most of the mentions for bananas in Wikipedia articles are on the topic of industry which are less likely to reveal the common sense properties of banana. To test this hypothesis, we experiment with a strategy based on topic models. Our strategy relies on the following steps:

1. Identify the topics which are most relevant for the target word w;
2. For each of the selected topics t, select sentences $s_1^t, ..., s_n^t$ mentioning w from documents that are closely related to this topic.

For each of the selected topics t, we can then use the sentences $s_1^t, ..., s_n^t$ to construct a topic-specific vector $\mathbf{w}^t$, using any of the strategies from Section 3.2.1. The final representation of w will be computed as a weighted average of these topic-specific vectors, as will be explained in Section 3.2.3.

We now explain these two steps in more detail. First, we use Latent Dirichlet Allocation (LDA) [13] to obtain a representation of each document d in the considered corpus as a multinomial distribution over m topics. Let us write $\tau_i(d)$ for the weight of topic i in the representation of document d, where $\sum_{i=1}^{m} \tau_i(d) = 1$. Suppose that the word

w is mentioned N_w times in the corpus, and let d_j^w be the document in which the j^{th} mention of w occurs. Then we define the importance of topic i for word w as follows:

$$\tau_i(w) = \frac{1}{N_w} \sum_{j=1}^{N_w} \tau_i(d_j^w) \tag{3.1}$$

In other words, the importance of topic i for word w is defined as the average importance of topic i for the documents in which w occurs. To select the set of topics $\mathcal{T}_w$ that are relevant to w, we rank the topics from most to least important and then select the smallest set of topics whose cumulative importance is at least 60%, i.e. $\mathcal{T}_w$ is the smallest set of topics such that $\sum_{t_i \in \mathcal{T}_w} \tau_i(w) \geq 0.6$. The reason for this manually selected threshold at 60% is to capture the majority of topics and avoid too many less relevant topics.

For each of the topics t_i in $\mathcal{T}_w$ we select the corresponding sentences $s_1^t, ..., s_n^t$ as follows. We rank all the documents in which w is mentioned according to $\tau_i(d)$. Then, starting with the document with the highest score (i.e. the document for which topic i is most important), we iterate over the ranked list of documents, selecting all sentences from these documents in which w is mentioned, until we have obtained a total of n sentences.

3.2.3 Combining Word Representations

Section 3.2.1 highlighted a number of strategies that could be used to construct a vector representation of a target word w. As mentioned before, it can be beneficial to combine vector representations from different transformer layers. To this end, we propose to learn a weighted average of the different input vectors, using a task specific supervision signal. In particular, let $\mathbf{w}_1, ..., \mathbf{w}_k$ be the different vector representations we have available for word w (e.g. the vectors from different transformer layers). To combine these vectors, we compute a weighted average as follows:

$$\lambda_i = \frac{\exp(a_i)}{\sum_{j=1}^{k} \exp(a_i)} \tag{3.2}$$

$$\mathbf{w} = \frac{\sum_i \lambda_i \mathbf{w}_i}{\| \sum_i \lambda_i \mathbf{w}_i \|} \tag{3.3}$$

where the scalar parameters $a_1, ...a_k \in \mathbb{R}$ are jointly learned with the model in which $\mathbf{w}$ is used. Another possibility would be to concatenate the input vectors $\mathbf{w}_1, ..., \mathbf{w}_k$. However, this significantly increases the dimensionality of the word representations, which can be challenging in downstream applications. In initial experiments, we also confirmed that this concatenation strategy indeed under-performs the use of weighted averages.

If topic-specific vectors are used, we also want to compute a weighted average of the available vectors. However, (3.2)–(3.3) cannot be used in this case, because the set of topics for which topic-specific vectors are available differs from word to word. Let us write $\mathbf{w}^i_{\mathit{topic}}$ for the representation of word w that was obtained for topic t_i, where we assume $\mathbf{w}^i_{\mathit{topic}} = \mathbf{0}$ if $t_i \notin \mathcal{T}_w$. We then define:

$$\mu^w_i = \frac{\exp(b_i) \cdot 1[t_i \in \mathcal{T}_w]}{\sum_{j=1}^{k} \exp(b_i) \cdot 1[t_j \in \mathcal{T}_w]} \tag{3.4}$$

$$\mathbf{w}_{\mathit{topic}} = \frac{\sum_i \mu^w_i \mathbf{w}^i_{\mathit{topic}}}{\| \sum_i \mu^w_i \mathbf{w}^i_{\mathit{topic}} \|} \tag{3.5}$$

where $1[t_i \in \mathcal{T}_w] = 1$ if topic t_i is considered to be relevant for word w (i.e. $t_i \in \mathcal{T}_w$), and $1[t_i \in \mathcal{T}_w] = 0$ otherwise. Note that the softmax function in (3.4) relies on the scalar parameters $b_1, ..., b_k \in \mathbb{R}$, which are independent of w. However, the softmax is selectively applied to those topics that are relevant to w, which is why the resulting weight μ^w_i is dependent on w, or more precisely, on the set of topics $\mathcal{T}_w$.

3.3 Evaluation

We compare the proposed strategy with standard word embeddings and existing CLM-based strategies. In Section 3.3.1 we first describe our experimental setup. Section 3.3.2 then provides an overview of the datasets we used for the experiments, where we focus on lexical classification benchmarks. These benchmarks in particular allow us to assess how well various semantic properties can be predicted from the word vectors. The experimental results are discussed in Section 3.3.3 and a qualitative analysis is presented in Section 3.3.4.

3.3.1 Experimental Setup

We experiment with a number of different strategies for obtaining word vectors:

C_{last}

$\mathbf{C}_{\textbf{last}}$ We take the vector representation of w from the last transformer layer (i.e. $\mathbf{w}^s_{12}$ or $\mathbf{w}^s_{24}$).

$\mathbf{C}_{\textbf{input}}$ We take the input embedding of w (i.e. $\mathbf{w}_0$).

$\mathbf{C}_{\textbf{avg}}$ We take the average of $\mathbf{w}_0, \mathbf{w}^s_1, ..., \mathbf{w}^s_{12}$ for the *base* models and $\mathbf{w}_0, \mathbf{w}^s_1, ..., \mathbf{w}^s_{24}$ for the *large* models.

$\mathbf{C}_{\textbf{all}}$ We use all of $\mathbf{w}_0, \mathbf{w}^s_1, ..., \mathbf{w}^s_{12}$ as input for the *base* models, and all of $\mathbf{w}_0, \mathbf{w}^s_1, ..., \mathbf{w}^s_{24}$ for the *large* models. These vectors are then aggregated using (3.2)–(3.3), i.e. we use a learned soft selection of the transformer layers.

$\mathbf{C}_{\textbf{mask}}$ We replace the target word by [MASK] and use the corresponding output vector.

For words consisting of more than one word-piece, we average the corresponding vectors in all cases, except for $\mathbf{C}_{\textbf{mask}}$ where we always end up with a single vector (i.e. we replace the entire word by a single [MASK] token). We also consider three variants that rely on topic-specific vectors:

T$_{\text{last}}$ We learn topic-specific vectors using the last transformer layers. These vectors are then used as input to (3.4)–(3.5).

T$_{\text{avg}}$ Similar to the previous case but using the average of all transformer layers.

T$_{\text{mask}}$ Similar to the previous cases but using the output vector of the masked word mention.

Furthermore, we consider variants of **T$_{\text{last}}$**, **T$_{\text{avg}}$** and **T$_{\text{mask}}$** in which a standard (i.e. unweighted) average of the available topic-specific vectors is computed, instead of relying on (3.4)–(3.5). We will refer to these averaging-based variants as **A$_{\text{last}}$**, **A$_{\text{avg}}$** and **A$_{\text{mask}}$**. As baselines, we also consider the two Word2vec models [73]:

SG 300-dimensional Skip-gram vectors trained on a May 2016 dump of the English Wikipedia, using a window size of 5 tokens, and minimum frequency threshold of 10.

CBOW 300-dimensional Continuous Bag-of-Words vectors trained on the same corpus and with the same hyperparameters as **SG**.

We show results for four pre-trained CLMs [**?** 69]: BERT-base-uncased, BERT-large-uncased, RoBERTa-base-uncased, RoBERTa-large-uncased[2]. As the corpus for sampling word mentions, we used the same Wikipedia dump as for training the word embeddings models. For **C$_{\text{mask}}$**, **C$_{\text{last}}$**, **C$_{\text{avg}}$** and **C$_{\text{all}}$** we selected 500 mentions. For the topic-specific strategies (**T$_{\text{last}}$**, **T$_{\text{avg}}$** and **T$_{\text{mask}}$**) we selected 100 mentions per topic. To obtain the topic assignments, we used Latent Dirichlet Allocation [13] with 25 topics. We set $\alpha = 0.0001$ to restrict the total number of topics attributed to a document, and use default values for the other hyper-parameters[3]. To select the relevant topics for a given word w, we find the smallest set of topics whose cumulative importance score $\tau_i(w)$ is

[2]We used the implementations from `https://github.com/huggingface/transformers`.

[3]We used the implementation from `https://radimrehurek.com/gensim/wiki.html`.

	BERT-base				**BERT-large**				**RoBERTa-base**				**RoBERTa-large**			
	MC	**CS**	**SS**	**BD**	**MC**	**CS**	**SS**	**BD**	**MC**	**CS**	**SS**	**BD**	**MC**	**CS**	**SS**	**BD**
SG	59.6	54.5	55.6	**49.1**	59.6	54.5	55.6	49.1	59.6	54.5	55.6	**49.1**	59.6	54.5	55.6	**49.1**
CBOW	61.1	50.6	48.4	45.0	61.1	50.6	48.4	45.0	61.1	50.6	48.4	45.0	61.1	50.6	48.4	45.0
$\mathbf{C}_{\mathbf{mask}}$	60.8	51.7	59.7	42.6	61.8	53.4	59.5	42.0	62.5	51.8	58.5	40.0	61.3	53.2	59.2	42.8
$\mathbf{C}_{\mathbf{last}}$	60.0	51.4	59.1	46.1	58.2	53.4	59.0	46.3	56.5	49.4	58.2	42.1	57.9	47.7	58.8	44.9
$\mathbf{C}_{\mathbf{input}}$	58.8	40.1	50.2	40.3	57.2	42.0	51.7	40.2	45.8	24.1	44.4	37.9	41.2	20.6	52.6	40.0
$\mathbf{C}_{\mathbf{avg}}$	59.9	49.6	59.1	44.2	60.0	47.1	58.9	43.5	55.7	40.7	50.2	41.3	59.5	47.4	58.8	43.9
$\mathbf{C}_{\mathbf{all}}$	59.9	51.2	59.5	46.4	61.7	50.7	58.4	42.6	45.3	39.3	52.6	36.7	48.2	40.2	56.6	40.4
$\mathbf{T}_{\mathbf{mask}}$	60.9	54.1	60.5	45.7	62.8	55.0	61.1	45.9	58.6	49.4	56.7	42.1	59.2	50.4	57.2	43.3
$\mathbf{T}_{\mathbf{last}}$	63.0	51.8	59.7	47.3	62.1	55.8	61.6	**49.2**	52.3	43.9	54.6	43.3	62.1	48.8	59.5	45.1
$\mathbf{T}_{\mathbf{avg}}$	61.0	52.7	59.6	43.4	**65.2**	54.8	**60.7**	48.4	54.5	39.9	55.9	41.5	59.5	47.4	60.0	45.2
$\mathbf{A}_{\mathbf{mask}}$	63.1	53.9	59.2	41.4	63.2	56.8	60.6	41.5	**64.0**	55.3	60.6	40.8	63.4	**57.3**	62.0	42.3
$\mathbf{A}_{\mathbf{last}}$	62.8	52.4	59.6	44.4	61.4	55.5	60.6	46.7	55.7	36.8	56.5	42. 7	59.6	47.8	59.7	44.5
$\mathbf{A}_{\mathbf{avg}}$	61.3	49.7	57.9	44.4	63.3	52.2	59.4	43.8	57.6	40.6	56.4	39.8	59.4	47.3	58.5	42.4
$\mathbf{C}_{\mathbf{mask}}$**-PCA**	61.8	52.6	58.8	41.2	62.3	53.2	60.1	41.6	61.5	52.6	59.2	40.3	62.2	51.5	59.1	40.5
$\mathbf{T}_{\mathbf{mask}}$**-PCA**	**63.3**	**56.2**	**62.6**	46.9	64.4	**57.3**	60.6	48.0	61.6	**55.8**	**62.5**	46.0	**65.4**	56.3	**64.1**	46.4

Table 3.1: Results of lexical feature classification experiments for the extended McRae feature norms (MC), CSLB norms (CS), WordNet Supersenses (SS) and BabelNet domains (BD). Results are reported in terms of F1 (%) **.**

at least 60%, with a maximum of 6 topics. In the experiments, we restrict the vocabulary to those words with at least 100 occurrences in Wikipedia.

3.3.2 Datasets

For the experiments, we focus on a number of lexical classification tasks, where categories of individual words need to be predicted. In particular, we used two datasets which are focused on commonsense properties (e.g. *dangerous*): the extension of the

Dataset	**Type**	**Words**	**Properties**
McRae	Commonsense	475	49
CSLB	Commonsense	570	54
WN supersenses	Taxonomic	24,324	24
BN domains	Topical	43,319	34

Table 3.2: Overview of the considered datasets.

McRae feature norms dataset [72] that was introduced by Forbes et al. [37][4] and the CSLB Concept Property Norms[5]. We furthermore used the WordNet supersenses dataset[6], which groups nouns into broad categories (e.g. *human*). Finally, we also used the BabelNet domains dataset[7] [19], which assigns lexical entities to thematic domains (e.g. *music*).

In our experiments, we have only considered properties/classes for which sufficient positive examples are available, i.e. at least 10 for McRae, 30 for CSLB, and 100 for WordNet supersenses and BabelNet domains. For the McRae dataset, we used the standard training-validation-test split. For the other datasets, we used random splits of 60% for training, 20% for tuning and 20% for testing. An overview of the datasets is shown in Table 3.2.

For all datasets, we consider a separate binary classification problem for each property and we report the (unweighted) average of the F1 scores for the different properties. To classify words, we feed their word vector directly to a sigmoid classification layer. We optimise the network using AdamW with a cross-entropy loss. The batch size and learning rate were tuned, with possible values chosen from 4,8,16 and 0.01, 0.005, 0.001, 0.0001 respectively. Note that for $\mathbf{C_{all}}$ and the topic-specific variants, the classification network jointly learns the parameters of the classification layer and the attention weights in (3.2) and (3.4) for combining the input vectors.

[4]`https://github.com/mbforbes/physical-commonsense`

[5]`https://cslb.psychol.cam.ac.uk/propnorms`

[6]`https://wordnet.princeton.edu/download`

[7]`http://lcl.uniroma1.it/babeldomains/`

3.3.3 Results

The results are shown in Table 3.1. Regarding vectors derived from BERT-base and BERT-large, we consistently see that the topic-specific variants outperform the different **C**-variants by a small margin. Regarding vectors derived from RoBERTa-base and RoBERTa-large, the performances of some topic-specific variants drops slightly while $\mathbf{T_{mask}}$**-PCA** and $\mathbf{T_{avg}}$ still outperform all the **C**-variants in most cases. This confirms our main hypothesis, namely that using topic models to determine how context sentences are selected has effects on the quality of the resulting word representations. Among the **C**-variants, the best results are obtained by $\mathbf{C_{mask}}$. None of the three **T**-variants consistently outperforms the others. Surprisingly, the **A**-variants outperform the corresponding **T**-variants in several cases. This suggests that the outperformance of the topic-specific vectors primarily comes from the fact that the context sentences for each word were sampled in a more balanced way (i.e. from documents covering a broader range of topics), rather than from the ability to adapt the topic weights based on the task. This is a clear benefit for applications, as the **A**-variants allow us to simply represent each word as a static word vector.

The performance of SG and CBOW is still competitive. In CSLB and BabelNet Domains, these traditional word embedding models outperform all of the **C**-variants. This seems to be related, at least in part, to the lower dimensionality of these vectors. The classification network has to be learned from a rather small number of examples, especially for McRae and CSLB. Having 768 or 1024 dimensional input vectors can be problematic in such cases. To analyse this effect, we used Principal Component Analysis (PCA) to reduce the dimensionality of the CLM-derived vectors to 300. For this experiment, we focused in particular on $\mathbf{C_{mask}}$ and $\mathbf{T_{mask}}$. The results are also shown in Table 3.1 as $\mathbf{C_{mask}}$**-PCA** and $\mathbf{T_{mask}}$**-PCA**. As can be seen, this dimensionality reduction step has a clearly beneficial effect, with $\mathbf{T_{mask}}$**-PCA** outperforming all baselines, except for the BabelNet domains benchmark.

It should be noted that these four evaluation benchmarks focus on different semantic properties, i.e. McRae and CSLB focus on commonsense properties, WN supersenses highlight taxonomic properties, whereas BN domains concentrate on topical properties. Comparing topic-specific vectors' performance on four benchmarks provides us more insight into what kind of semantic properties they capture. The improvements brought by topic-specific representations over **C**-variants on BN domains are more significant than the other three datasets. The poor performance of **C**-variants on BN domains shows they struggle with thematic properties. On the other hand, we find that topic-specific vectors consistently outperform the **C**-variants on BN domains. This suggests that our proposed Topic-aware mentions selection strategy makes word representation capture more topical properties. We can also find the improved performance of topic-specific vectors on McRae and CSLB, suggesting that our topic-aware mentions selection also helps capture commonsense properties. The improvement brought to WN supersenses benchmark is less significant as **C**-mask already obtains decent scores. We assume that **C**-mask strategy forces the CLMs to prioritize the taxonomic properties modelling [67] so that our topic-aware mention strategy has limited value to add to it.

3.3.4 Qualitative Analysis

Topic-specific vectors can be expected to focus on different properties, depending on the chosen topic. In this section, we present a qualitative analysis in support of this view. In Table 3.3 we list, for a sample of words from the WordNet supersenses dataset, the top 5 nearest neighbours per topic in terms of cosine similarity. For this analysis, we used the BERT-base masked embeddings. We can see that for the word *'partner'*, its topic-specific embeddings correspond to its usage in the context of 'finance', 'stock market' and 'fiction'. These three embeddings roughly correspond to three different senses of the word[8]. This de-conflation or implicit disambiguation is also found for

[8]In fact, we can directly pinpoint these vectors to the following WordNet [75] senses: `partner.n.03`, `collaborator.n.03` and `spouse.n.01`.

WORD	TOPIC	NEAREST NEIGHBOURS
partner	{research, professor, science, education}	beneficiary, creditor, investor, employer, stockholder
	{football, republican, coach, senate}	lobbyist, bookkeeper, cashier, stockbroker, clerk
	{game, book, novel, story}	nanny, spouse, lover, friend,secretary
cell	{protein, disease, medical, cancer}	lymphocyte, macrophage, axon, astrocyte, organelle
	{food, plant, water, gas, power}	electrode, electrolyte, cathode, anode, substrate
	{physics, mathematics, space, theory}	surface, torus, mesh, grid, cone
port	{station, building, railway, historic}	harbor, seaport, dock, waterfront, city
	{radio, station, fm, software, data}	link, gateway, router, line, socket
	{game, book, novel, story}	version, remake, compilation, patch, modification
bulb	{station, building, railway, historic}	lamp, transformer, dynamo, projector, lighting
	{protein, disease, medical, cancer}	epithelium, ganglion, nucleus, gland, cortex
	{species, genus, described, flowers}	rootstock, fern, vine, tuber, clover
mail	{station, building, railway, historic}	cargo, grain, baggage, coal, livestock
	{game, book, novel, story}	paper, jewelry, telephone, telegraph, typewriter
	{party, election, minister, elected}	telemarketing, spam, wiretap, internet, money
fingerprint	{radio, station, fm, software}	signature, checksum, bitmap, texture, text
	{game, book, novel, story}	cadaver, skull, wiretap, body, tooth
	{party, election, minister, elected}	wiretap, forensics, postmortem, polygraph, check
sky	{greek, ancient, castle, king}	underworld, sun, afterlife, zodiac, moon
	{river, lake, mountain, island}	horizon, ocean, earth, sun, globe
	{physics, mathematics, space, theory}	ionosphere, sun, globe, earth, heliosphere
strength	{food, plant, water, gas}	stiffness, ductility, hardness, permeability, viscosity
	{game, book, novel, story}	intelligence, agility, charisma, power, telepathy
	{army, regiment, navy, ship}	morale, firepower, resistance, force, garrison
noon	{physics, mathematics, space, theory}	declination, night, equinox, perihelion, latitude
	{army, regiment, navy, ship}	dawn, sunset, night, morning, shore
galaxy	{physics, mathematics, space, theory}	nebula, quasar, pulsar, nova, star
	{game, book, novel, story}	globe, future, world, planet, nation

Table 3.3: Nearest neighbours of topic-specific embeddings for a sample of words from the WordNet SuperSenses dataset, using BERT-base embeddings. The top 6 selected samples illustrate clear topic distributions per word sense, and the bottom 4 also show topical properties within the same sense. The most relevant words for each topic are shown under the column TOPIC.

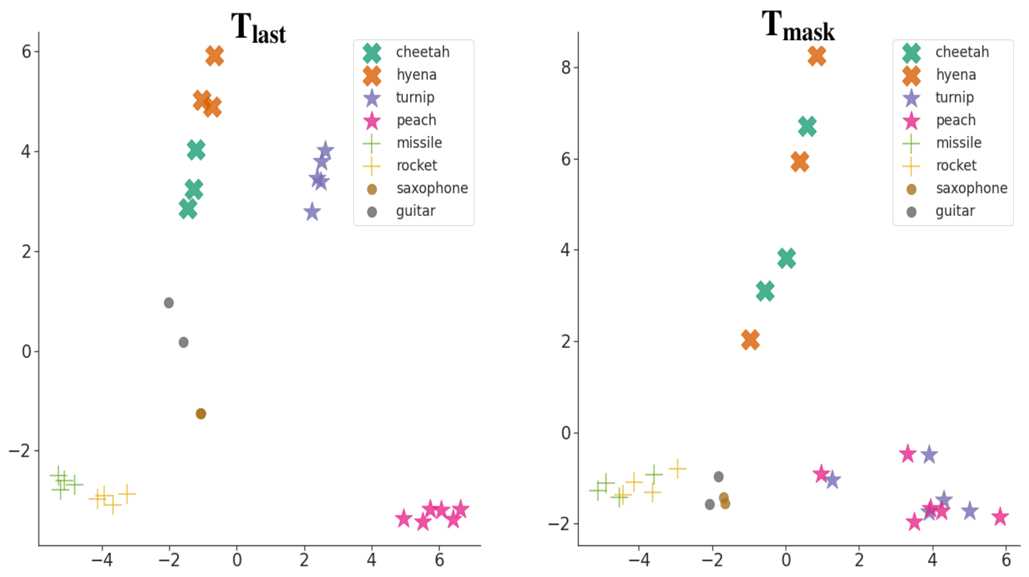

Figure 3.1: BERT-base topic-specific vectors when using the output vectors without using masking (left) and with masking (right). Words have been selected from the McRae dataset.

words such as *‘cell’*, *‘port’*, *‘bulb’* or *‘mail’*, which shows a striking relevance of the role of mail in the election topic, being semantically similar in the corresponding vector space to words such as ‘telemarketing’, ‘spam’ or ‘wiretap’. In the case of *‘fingerprint’*, we can also see some implicit disambiguation (distinguishing between fingerprinting in computer science, as a form of hashing, and the more traditional sense). However, we also see a more topical distinction, revealing differences between the role played by fingerprints in fictional works and forensic research. This tendency of capturing different contexts is more evidently shown in the last four examples. First, for *‘sky’* and *‘strength’*, the topic-wise embeddings do *not represent different senses of these words*, but rather indicate different types of usage (possibly related to cultural or commonsense properties). Specifically, we see that the same sense of *‘sky’* is used in mythological, landscaping and geological contexts. Likewise, *‘strength’* is clustered into different mentions, but while this word also preserves the same sense, it is clearly used in different contexts: physical, as a human feature, and in military contexts. Finally, *‘noon’* and *'galaxy'* (which only occur in two topics), also show this topicality. In both cases, we have representations that reflect their physics and everyday usages, for the same senses of these words.

As a final analysis, In Figure 3.1 we plot a two-dimensional PCA-reduced visualization of selected words from the McRae dataset, using two versions of the topic-specific vectors: $\mathbf{T_{mask}}$ and $\mathbf{T_{last}}$. In both cases, BERT-base was used to obtain the vectors. We select four pairs of concepts which are topically related, which we plot with the same datapoint marker (animals, plants, weapons and musical instruments). For $\mathbf{T_{last}}$, we can see that the different topic-specific representations of the same word are clustered together, which is in accordance with the findings from Ethayarajh [31]. For $\mathbf{T_{mask}}$, we can see that the representations of words with similar properties (e.g. *cheetah* and *hyena*) become more similar, suggesting that $\mathbf{T_{mask}}$ is more tailored towards modelling the semantic properties of words, perhaps at the expense of a reduced ability to differentiate between closely related words. The case of *turnip* and *peach* is particularly striking, as the vectors are clearly separated in the $\mathbf{T_{last}}$ plot, while being clustered together in the $\mathbf{T_{mask}}$ plot.

Given the strength of $\mathbf{T_{mask}}$ shown in this qualitative analysis, it's intriguing that they fail to give impressive results on the lexical classification tasks. We assume the issue might originate from the vectors merging process when the topic-specific vectors are averaged to generate a single vector per word. Taking the average of several topic vectors will likely lose some sensitive semantic information. Although we try to alleviate the issue by averaging topic-specific vectors based on the learnt topic weight and expect the most relevant topics to be prioritized, our proposed neural network classifier can only learn fixed weights for all topics regardless of the activation word. We assume this is the obstacle that makes $\mathbf{T_{mask}}$ fail to fulfil its expected function and realize its full potential. In principle, this issue can be overcome by learning topic weights for each word, i.e., the industry topic has high weights for *strike* and low weights for *banana*. Due to the time limitation, this will not be further implemented in this thesis and will be left for future work.

3.4 Summary

We have proposed a strategy for learning static word vectors, in which topic models are used to help select diverse mentions of a given target word and a contextualized language model is subsequently used to infer vector representations from the selected mentions. We found that selecting an equal number of mentions per topic outperforms purely random selection strategies, even though the improvements on some benchmarks are limited.

We also considered the possibility of learning a weighted average of topic-specific vector representations, which in principle should allow us to "tune" word representations to different tasks, by learning task-specific topic importance weights. However, in practice we found that a standard average of the topic specific vectors leads to a comparable performance, suggesting that the outperformance of our vectors comes from the fact that they are obtained from a more diverse set of contexts. Motivated by this discovery, the research of next chapter aims at obtaining static word embedding efficiently by exploring more strategies of mention selection.

Chapter 4

Exploring Other Mention Selection Strategies

4.1 Introduction

The main aim of this chapter is to answer the second research question: can higher-quality word embeddings be obtained from a few mentions of each word? This research question is motivated by the practical desire to distil word vectors from language models in a more efficient way. We aim to answer this questions by empirically analyzing a range of strategies for selecting mentions of a given word w. By using those proposed sentence selection strategies, we expect that a word representation obtained from a few mentions will be comparable with or even stronger than its counterpart obtained from a large number of mentions. Furthermore, comparing the effectiveness of different sentence selection strategies can also provide us with insights into how language models acquire knowledge about word meaning.[1]:

Specifically, we aim to answer the following questions with our analysis.

- Can we distil high-quality word vectors from language models given a limited number of mentions for each target word?

[1]All code and data to replicate our experiments is available at `https://github.com/Activeyixiao/Sentence-Selection-Strategies/`.

- Does masking the target word still lead to better results in such a setting?
- Can we effectively predict which sentences are most likely to be useful for learning the meaning of a given word?

4.2 Distilling Word Embeddings

To obtain the vector representation of a word w, we first sample n sentences $S_1, ..., S_n$ mentioning w. Unless noted otherwise, the source corpus from which these sentences are sampled in our experiments is always Wikipedia, specifically a dump from March 2021. Wikipedia has been used extensively in many areas of NLP, with notable use cases including lexical semantics [78], knowledge extraction and management, or taxonomy learning [97]. For our purposes, moreover, Wikipedia is also a clean resource for encyclopedic information, which, despite its collective nature, undergoes strict editorial revisions, and which has a particular structure that we can exploit. We now discuss the process of sampling our target sentences from Wikipedia. Each of the sentences $S_1, ..., S_n$ is fed through a masked language model such as BERT [**?**] or RoBERTa [69]. From each sentence S_i we obtain a contextualised vector $\mathbf{w_i}$ using one of the following alternatives that was already explained in the Chapter 3:

- MASK
- LAST
- AVG

Same as the method used in the Chapter 3, the final embedding $\mathbf{w}$ of word w is obtained by averaging the contextualised vector $\mathbf{w_1}, ..., \mathbf{w_n}$.

4.3 Sentence Selection Strategies

We need a strategy for selecting n sentences that mention a given target word w. Our baseline strategy, which we will refer to as RAND, is to randomly sample different sentences from Wikipedia. To avoid poorly structured sentences, we avoid sentences with more than 60 or fewer than 7 words. We now discuss a number of alternative sentence selection strategies, aimed at providing us with more informative sentences. Our hypothesis is that this will allow us to obtain word vectors capturing more semantic properties from a small number of sentences, which is essential for scaling up the methods for distilling static word embeddings from language models. Given this focus on efficiency, we are particularly interested in sentence selection strategies with a low computational overhead. We first consider two strategies that rely on the structure of Wikipedia:

- HOME: If there is a Wikipedia article about w, we select the first n sentences mentioning w from that article. If w does not have a Wikipedia article, we fall back on RAND. An example sentence mentioning *banana* obtained by this strategy is: "A banana is an elongated, edible fruit – botanically a berry – produced by several kinds of large herbaceous flowering plants in the genus Musa"

- INTRO: We only sample sentences that occur in the introductory section of a Wikipedia article, regardless of what the article is about. For example, if *apple* is mentioned in the introduction section of the Wikipedia article for *fruit*, that sentence can be selected to model the semantic properties of *apple*. Noting that the Wikipedia style guide[2] states that introductory sections should be “written in a clear, accessible style with a neutral point of view”, this strategy seems to be a good fit with our requirement of retrieving sentences where core properties of words are more likely to be mentioned explicitly. An example sentence men-

[2]`https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Lead_section`

tioning *banana* obtained by this strategy is: "The area produces citrus, olives, tomatoes and market-garden vegetables, and is one of the few parts of Europe where commercial banana production is possible".

We also analyse a number of strategies that rely on aspects of the sentences themselves:

- START: We only sample sentences which start with the word w in plural form. The intuition is that such sentences are likely to express generic knowledge about w. An example sentence mentioning *banana* obtained by this strategy is "Bananas must be transported over long distances from the tropics to world markets".
- ENUM: We first select all sentences in which w is preceded or succeeded by a comma or the word 'and'. Then we rank these sentences based on the number of commas, as a simple strategy for prioritizing longer enumerations, and we select the n highest ranked sentences. The intuition is that enumerations can provide us with useful knowledge, capturing the fact that the words in the enumeration have some property in common with w. An example sentence mentioning *banana* obtained by this strategy is: "There are also wild relatives of jackfruit, mango, cardamom, turmeric and banana".
- PMI: For all words that co-occur with w in at least 2 sentences, we compute their Pointwise Mutual Information (PMI) in an offline preprocessing step. This PMI score reflects to what extent these words appear more often in the same sentence than would be expected by chance, given their overall frequency. Given a target word w, we first identify the n words whose PMI score with w is highest. For each of these n related words, we then randomly select one sentence mentioning both words. For example, *banana* has a high PMI score with *fruit*. A selected sentence should mention both words *banana* and *fruit*. An example is: "The common fruits that are used in the preparation include banana, apple, kiwi, strawberry, papaya, pineapple, mango, and soursop".

- DEF: We extract the (primary) definition of w from the English fragment of Wiktionary[3]. This is a free-content multilingual dictionary that describes words of many languages using definitions and descriptions in English. An example definition of *banana* is "Banana is an elongated, curved tropical fruit that grows in bunches and has a creamy flesh and a smooth skin".
- GENERIC: We also consider using sentences from GenericsKB [12], a large-scale resource containing naturally occurring generic sentences originating from a text corpus or knowledge graph triples. We only select sentences in GenericsKB from a text corpus because the sentences generated from knowledge graph triples tend to be short and artificial. We rank the sentences for a target word based on their confidence score in GenericsKB and select the top n. An example sentence is: "Bananas have no fat, cholesterol or sodium".

For all strategies, if there are fewer than n sentences that can be selected, we fall back to RAND for the remaining sentences.

4.4 Experiments

In this section, we empirically compare the sentence selection strategies from Section 4.3.

Datasets We focus on the problem of predicting semantic properties of words. The reason is that in applications such as zero-shot learning or ontology completion, what matters is whether the word vectors capture particular properties. Following the evaluation method in Chapter 3, we use the four benchmarks and the same evaluation method because those four datasets evaluate word representations based on their capacity to capture different types of semantic properties. As mentioned in the last chapter

[3]`https://www.wiktionary.org`

		McR				CSLB				WNSS				BND			
		1	5	10	20	1	5	10	20	1	5	10	20	1	5	10	20
Mask	Rand	44.8	57.0	59.8	61.5	31.0	47.4	51.7	53.8	39.3	53.6	56.0	59.1	28.0	36.2	38.0	40.0
	Intro	44.0	57.9	58.7	60.7	34.4	47.3	50.8	53.8	41.6	54.2	56.5	57.6	28.3	36.6	38.5	40.0
	Home	55.3	59.2	61.9	60.0	42.0	$\underline{50.4}$	53.2	54.5	45.9	$\underline{55.2}$	58.2	58.7	28.9	35.8	37.4	39.1
	Start	42.1	51.6	54.7	56.8	29.5	44.8	47.4	50.1	38.1	51.0	54.9	56.4	28.3	35.7	38.3	39.9
	Enum	42.9	53.9	55.5	57.1	29.9	43.3	47.8	44.6	41.0	52.5	54.8	56.2	28.3	36.1	39.3	40.0
	PMI	56.8	57.0	59.2	61.6	48.9	46.1	**54.0**	54.4	43.1	55.1	58.3	58.6	$\underline{29.5}$	$\underline{37.6}$	$\underline{39.7}$	$\underline{41.0}$
	Generic	46.7	52.5	55.4	57.0	33.9	45.7	47.8	50.4	36.4	51.3	55.3	57.8	26.0	34.4	36.7	38.8
	Def+Home	$\underline{56.9}$	$\underline{59.9}$	$\underline{62.2}$	**64.1**	$\underline{49.6}$	$\underline{50.4}$	53.2	**56.0**	49.6	$\underline{55.2}$	$\underline{58.5}$	$\underline{59.3}$	29.2	35.7	37.4	39.1
	Def+Rand	55.6	58.2	59.2	62.6	48.8	49.8	51.8	55.5	$\underline{50.3}$	$\underline{55.2}$	57.1	58.6	29.3	35.7	37.9	39.4
Last	Rand	55.5	59.0	62.3	61.6	46.1	48.3	53.9	53.5	49.4	56.5	58.0	59.0	35.4	$\underline{42.9}$	44.7	45.7
	Intro	53.4	58.7	61.5	59.8	43.3	48.8	50.1	51.8	50.2	**58.3**	58.0	59.1	35.8	42.8	**44.8**	45.6
	Home	$\underline{58.3}$	$\underline{61.8}$	$\underline{62.6}$	63.0	47.8	48.7	51.8	51.0	52.0	**58.3**	59.1	**59.6**	35.7	42.3	43.9	44.9
	Start	53.7	59.5	58.9	59.8	43.4	52.8	53.2	$\underline{55.3}$	45.4	55.4	57.5	58.6	32.6	38.7	40.5	41.5
	Enum	47.4	59.8	58.1	60.0	41.9	47.8	47.3	52.5	49.5	55.3	57.0	57.4	35.3	42.7	43.7	45.4
	PMI	55.2	60.0	61.8	$\underline{63.4}$	43.9	**53.0**	**54.0**	54.4	49.7	57.0	59.1	**59.6**	36.6	42.2	44.6	**45.8**
	Generic	54.3	60.7	59.8	61.1	45.1	49.2	51.2	51.3	50.3	57.3	57.9	58.9	36.1	42.3	43.2	44.3
	Def+Home	57.0	60.4	61.6	63.0	**50.1**	48.4	52.5	51.7	$\underline{55.2}$	**58.3**	**59.6**	59.4	$\underline{37.2}$	42.4	44.0	45.1
	Def+Rand	57.6	60.5	58.8	61.9	**50.1**	49.1	51.5	52.9	$\underline{55.2}$	58.0	59.4	59.1	$\underline{37.2}$	41.7	44.1	45.5
Avg	Rand	56.5	**62.7**	61.2	60.8	45.4	49.5	50.0	49.1	53.0	56.8	57.7	57.5	39.4	43.5	44.4	$\underline{45.1}$
	Intro	57.4	60.1	58.3	58.8	44.4	49.2	49.2	48.0	52.8	$\underline{57.9}$	58.3	58.7	38.7	**43.7**	44.4	44.9
	Home	**59.4**	60.1	61.1	60.9	47.8	50.3	49.1	48.4	53.7	57.5	58.1	58.5	39.3	43.2	43.6	44.2
	Start	55.2	61.6	60.2	60.2	45.8	$\underline{50.5}$	$\underline{50.6}$	$\underline{51.8}$	47.5	55.0	57.6	58.0	34.8	39.7	41.0	41.4
	Enum	54.7	59.9	57.8	60.6	43.8	48.1	48.5	48.4	52.1	55.6	56.9	56.8	39.0	42.8	44.0	44.5
	PMI	58.5	61.7	**63.2**	$\underline{62.2}$	45.1	$\underline{50.5}$	50.5	50.4	53.2	57.8	$\underline{59.3}$	58.6	39.5	43.2	44.3	44.6
	Generic	58.7	61.0	60.1	61.5	42.3	44.6	46.1	46.1	52.6	56.7	57.6	57.7	39.1	43.1	43.4	44.0
	Def+Home	57.9	60.3	61.5	59.7	$\underline{49.6}$	49.3	48.1	46.5	**57.6**	57.3	58.7	$\underline{58.9}$	**40.4**	43.1	43.7	43.8
	Def+Rand	58.0	60.2	61.3	60.5	49.7	49.6	50.5	51.1	**57.6**	57.0	58.1	58.1	**40.4**	43.4	$\underline{44.5}$	44.8

Table 4.1: Results for word classification in terms of F1 score. Results were obtained using BERT-base. We report results for 1, 5, 10 and 20 sentences. The best results for a given benchmark and number of sentences are shown in bold. The best results within each embedding strategy (i.e. Mask, Last, Avg) are underlined.

of this thesis, McRae and CSLB focus on commonsense properties, WN supersenses highlight taxonomic properties, whereas BN domains concentrate on topical properties. As a downstream task, we also consider the ontology completion benchmark from [67]. In this case, word vectors are used as input features to a graph neural network, whose structure is determined by a given ontology or rule base. In particular, given a rule template such as $\star(x) \wedge \mathit{LocatedIn}(x,y) \rightarrow \mathit{CapitalCity}(y)$, the task is to predict which concepts can be used for the placeholder $\star$ to make the rule plausible.

Experimental Settings For word classification, the experimental settings is same as the experimental setting in Chapter 3. For ontology completion, we follow the same evaluation methodology as [67], which restricts the evaluation to concept names that appear at least twice in Wikipedia. We use the same hyperparameter settings, and we apply SVD to reduce the dimensionality of the word vectors to 300, as also suggested by [67]. For the pre-trained language models, we used the implementations from `https://github.com/huggingface/transformers`.

Results The results of the word classification experiments are summarized in Table 4.1. For these results, we used BERT-base-uncased; results for other language models will be discussed below. For the hybrid strategy DEF+HOME we select one sentence using DEF and the remaining sentences using HOME, and similar for DEF+RAND. Our main findings can be summarised as follows. First, compared to MASK, we find that LAST and AVG are far less sensitive to the sentence selection strategy and the number of sentences. Second, the best results are obtained with MASK in the case of MCR and CSLB and with LAST in the case of WNSS and BND. Third, RAND is remarkably competitive, with START, GENERIC, and ENUM underperforming RAND, while INTRO performs broadly similar. Furthermore, we found that MCR and CSLB are more sensitive to our sentence selection strategies than WNSS and BND, which suggests our sentence selection strategies improve word representations by capturing more commonsense semantic properties than taxonomic and topic properties. Overall the best

	Wine	**Econ**	**Olym**	**Tran**	**SUMO**
RAND	16.6	17.2	13.6	8.7	35.2
HOME	18.1	17.9	14.3	9.5	37.9
PMI	16.9	17.6	13.9	8.7	38.6
DEF+HOME	20.1	18.1	16.8	10.0	39.2
BERT-500	23.0	20.0	16.9	11.5	41.4

Table 4.2: Results for the ontology completion experiment (F1 score). Results were obtained for 20 sentences using BERT-base with the MASK strategy. Wine, Economy, Olympics and Transport are domain-specific ontologies; SUMO is a large open-domain ontology.

results are obtained with PMI, HOME, DEF+HOME and DEF+RAND, all of which clearly outperform RAND. The similar performance of DEF+HOME and DEF+RAND shows that the presence of the definition plays a critical role. Moreover, note that the first sentence selected by HOME is typically a definition as well (i.e. the first sentence of the Wikipedia article). For MASK, we clearly see that DEF+HOME outperforms HOME and that DEF+RAND outperforms RAND, while for LAST and AVG the advantage of adding the definition is less obvious.

The results for ontology completion are shown in Table 4.2. We find that HOME, PMI and DEF+HOME outperform RAND in almost all cases, with DEF+HOME performing particularly well. We furthermore note that these results approach the values that were reported by [67] with 500 randomly selected sentences, which are shown as BERT-500 in Table 4.2.

Statistical Significance Computation As the above results include many dimensions and comparisons, computing statistical significance can help us to know which increases or decreases in performance are meaningful. In Table 4.3, we compare each of our proposed strategies with RAND and compute the p value using Wilcoxon signed-rank test[116]. Since there are seven methods are computed, Bonferroni cor-

		McR				**CSLB**				**WNSS**				**BND**			
		1	5	10	20	1	5	10	20	1	5	10	20	1	5	10	20
MASK	INTRO	47.35	83.55	97.7	55.88	96.6	49.36	89.78	81.99	41.07	39.57	67.2	93.68	8.25	35.9	67.02	65.5
	HOME	**0.008**	19.94	44.28	68.16	**0.002**	5.50	84.21	15.67	**0.01**	14.84	11.34	41.07	11.21	9.31	3.82	7.69
	START	7.10	**0.27**	0.66	1.80	2.60	68.20	17.23	39.20	2.95	**0.46**	3.18	10.13	45.11	48.12	40.81	4.56
	ENUM	12.23	5.7	2.62	**0.04**	16.48	7.19	14.44	0.77	57.82	20.02	8.01	8.01	2.59	16.88	26.53	1.01
	PMI	44.2	87.74	51.14	53.34	23.9	47.11	47.2	22.63	1.05	1.59	15.63	0.80	0.92	1.55	4.93	5.32
	GENERIC	25.3	1.97	11.56	**0.61**	58.45	95.05	9.05	65.48	11.9	1.70	13.364	54.00	0.89	2.59	**0.68**	2.81
	DEF+HOME	**0.50**	**0.10**	71.65	35.17	**0.01**	**0.10**	2.00	**0.14**	**0.001**	**0.003**	1.05	4.21	1.75	83.13	44.13	60.01

Table 4.3: Results of statistic significance tests for lexical classification. We test each selection strategy against RAND and report the p values ($\%$). The statistically significant p value over critical value (after apply Bonferroni correction) in each benchmark are shown in bold.

	BERT-LARGE				ROBERTA-BASE				ROBERTA-LARGE			
	McR	CSLB	WNSS	BND	McR	CSLB	WNSS	BND	McR	CSLB	WNSS	BND
RAND	62.2	55.6	59.4	39.6	59.8	51.6	57.9	39.0	61.3	55.0	59.5	40.3
HOME	63.2	54.8	59.8	39.0	59.3	48.2	58.0	38.4	61.4	53.4	60.3	40.0
PMI	**65.0**	55.4	59.7	**41.3**	**63.6**	**54.0**	**58.7**	**39.8**	62.7	**56.0**	**60.1**	**44.1**
DEF+HOME	62.9	**56.8**	**59.9**	39.1	61.2	50.5	58.5	39.0	**63.1**	53.4	60.0	39.8

Table 4.4: Comparison of different language models for the word classification benchmarks. Results are reported in terms of F1 score. All word representations are obtained from 20 sentences with MASK strategy .

rection is applied so that the critical p value is $0.05/7 \approx 0.007$. The p value of each strategy below this value is considered statistically significant. It is noticeable that our well-performed strategies such as DEF+HOME are more statistically significant when the number of mention is small.

Comparison with Other Language Models In Table 4.4, we present results for BERT-large-uncased, RoBERTa-base and RoBERTa-large, to complement the results for BERT-base-uncased from Table 4.1. In accordance with the findings for BERT-

	McR	CSLB	WNSS	BND
BERT-BASE 500	60.8	51.7	58.3	42.6
BERT-LARGE 500	**62.2**	51.9	**60.2**	43.0
RoBERTa-BASE 500	61.8	49.7	58.2	40.8
RoBERTa-LARGE 500	60.3	54.0	60.0	42.5
SKIP-GRAM	59.6	**54.5**	55.6	**49.1**
CBOW	61.1	50.6	48.4	45.0

Table 4.5: Comparison with the MASK strategy when using 500 randomly sampled sentences, as well as with static embedding baselines.

base, we can see that the PMI strategy is highly effective, consistently outperforming RAND. The HOME and DEF+HOME strategies are somewhat less effective in these cases, especially for the RoBERTa models.

In Table 4.5 we present results from [67] whose vectors were obtained from 500 randomly sampled sentences using the MASK strategy, covering four language models: BERT-base-uncased, BERT-large-uncased, RoBERTa-base and RoBERTa-large. We find that the results with 20 sentences from Table 4.4 outperform these vectors (for McR and CSLB) or are at least competitive with them (for WNSS and BND), thus further illustrating the effectiveness of the sentence selection strategies. Table 4.5 also shows results for traditional static word vectors that were trained with Word2Vec. In particular, SKIP-GRAM and CBOW vectors were trained on the same Wikipedia dump that we used for sampling sentences (enwiki-20210320). We used a window size of 5 and a minimum frequency threshold of 10. Somewhat surprisingly, perhaps, the best overall results for BD are obtained with the SKIP-GRAM vectors. This provides further evidence for the observation from [67] that BERT-based vectors are particularly suitable for capturing taxonomic properties, while struggling with looser forms of semantic relatedness. For the McRae dataset, CBOW achieves the best results in Table 4.5, but without outperforming the best configurations from Table 4.4. The comparatively strong performance of SKIP-GRAM and CBOW for the McR and CSLB datasets

	McR	CSLB	WNSS	BND
Wikipedia	**61.5**	**53.8**	**59.1**	**40.0**
Books3	61.0	52.8	58.2	37.1
OpenSubtitles	58.3	45.8	51.9	35.5

Table 4.6: Comparison for the MASK strategy when using 20 randomly sampled sentences from Wikipedia, Books3, and OpenSubstitles.

may also be explained by the relatively small size of these datasets, which means that the higher dimensionality of the BERT-based vectors can be sub-optimal.

Comparison with Mention Selections from Other Corpora Besides the Wikipedia corpus we also consider two other corpora. The first corpus is *Books3*, a large dataset containing 196640 books derived from a copy of the contents of the Bibliotik [4]. The second corpus is *OpenSubtitles*, an English language corpus of subtitles from movies and television shows gathered by [103]. We select these two corpora because they provide an important source of natural use of the English language, which may prove helpful for sampling high-quality mentions of each word. In order to compare the effect of mention selection from different corpora, we adopt our baseline strategy RAND to select 20 random sentences for each word from each corpus. Then, following the MASK strategy described at the beginning of this chapter, we obtain static word embeddings from each corpus. Finally, these word embeddings are evaluated on the lexical classification task. In Table 4.6, we present evaluation results from which we found that *Wikipedia* outperforms *Books3* and *OpenSubtitles* in all four benchmarks. In contrast, *OpenSubtitles* underperforms in all datasets. Given that the texts in *Wikipedia* tend to be more informative than the texts in *Books3* and *OpenSubtitles*, this result suggests that selecting from Wikipedia is more likely to obtain informative mentions of each word.

[4]`https://twitter.com/theshawwn/status/1320282149329784833`

Experiment without padding random mentions As mentioned before, if fewer than n sentences can be selected for each selection strategy, we fall back to RAND for the remaining sentences. However, the number of random sentences might be too large for many words and and this might prevent us from observing the effect of some selection strategies. Therefore, we want to eliminate RAND's effect and measure the actual benefit of using our proposed selection strategies. Instead of using 20 mentions for each word, we select the minimum number of mentions that all proposed mention selection strategies can obtain. However, we found the minimum number of mentions obtained by all our proposed strategies is 1. For example, HOME has 993 words with only one mention; START even has 4627 words with only one mention. Since the minimum mention in all strategies is 1, the experiment under one mention, as previously reported in table 4.1, can eliminate the effort of RAND. As discussed in previous paragraph, HOME and DEF outperform RAND consistently while START and ENUM are underperforming.

Impact of Word Frequency While SKIP-GRAM and CBOW were found to be surprisingly competitive in the main experiments, these traditional word embedding models struggle with words that are relatively rare. In such cases, we can expect that the improved ability of language models to model sentence context would become more important. To test this hypothesis, we grouped all the words from the test sets of WNSS and BD into 6 splits, based on their number of occurrences n_{occ} in Wikipedia: $n_{occ} \leq 20$; $20 < n_{occ} <= 50$; $50 < n_{occ} <= 100$; $100 < n_{occ} <= 300$; $300 < n_{occ} <= 500$; $n_{occ} > 500$. The F1 score for each of these cases is reported in Table 4.7. Note that we did not consider McR and CSLB for this analysis, as they almost exclusively consist of frequent words. We also did not consider other mention selection strategies given the limited mentions meeting the selection criteria, i.e. we cannot find 500 mentions of *banana* for HOME. The results in Table 4.7 clearly show that the BERT-based vectors outperform SKIP-GRAM and CBOW for rare words. For BD, where SKIP-GRAM obtained the best overall results, we can see that the BERT-based

WordNet Supersenses						
	0-20	21-50	51-100	101-300	301-500	>500
Rand	**37.7**	**45.9**	**53.4**	**54.8**	**51.7**	**55.9**
CBOW	26.2	37.3	34.1	43.1	44.1	52.3
Skip-gram	30.3	41.8	42.3	47	51.6	54.3
BabelNet Domains						
	0-20	21-50	51-100	101-300	301-500	>500
Rand	**27.8**	**30.3**	**31.5**	**32.6**	34.2	40.1
CBOW	22.1	26.8	22.5	28.4	30.4	**50.8**
Skip-gram	19.6	28.8	31.1	31.8	**37.3**	50.2

Table 4.7: Breakdown of results for WNSS and BND in function of word frequency (F1 score). For relatively rare words (up to 300 occurrences in Wikipedia), the Rand strategy clearly outperforms the CBOW and Skip-gram baselines. For higher-frequency words, these baselines outperform Rand in the case of BD.

vectors outperform for words occurring up to 300 times in Wikipedia. In contrast, for the case of BD, for words that occur more than 500 times, CBOW and Skip-gram performs much better than Rand

Impact of Sentence Length When analyzing the disappointing performance of Generic, we observed that the sentences from GenericsKB tend to be very short. In Table 4.8 we therefore analyze the effect of sentence length. While there is no clear overall relationship between sentence length and performance, when selecting (Wikipedia) sentences consisting of 7 to 15 tokens, the performance noticeably drops. Many of the sentences from GenericsKB fall within this range, which suggests that the underperformance of Generic may be related to the short length of the selected sentences.

Qualitative Analysis In Tables 4.9, 4.10, 4.11, 4.12, 4.13, and 4.14 we present the top-5 sentences that were selected for the words “banana","falcon", and "bucket", for

	McR	CSLB	WNSS	BND
7-15 words	57.8	51.5	56.8	39.1
15-25 words	59.4	**53.6**	57.9	39.5
25-35 words	**64.0**	50.1	58.1	39.4
35-45 words	60.0	53.3	57.2	**40.1**
45-55 words	60.3	51.0	**58.3**	**40.1**

Table 4.8: Impact of sentence length on the quality of the resulting word vectors. All results were obtained with BERT-base, using the MASK strategy with 20 sentences .

the different strategies considered in this paper. These examples illustrate some of the strengths and weaknesses of these strategies. For example, the 5 RAND sentences for “banana" largely convey information that is irrelevant for learning the meaning of this word. However, since they sometimes use “banana” in an idiosyncratic way, BERT may be able to predict that the masked word is banana, which may result in vectors that behave more like those from the LAST strategy. Meanwhile, the sentences selected using HOME and PMI are typically more informative (e.g. describing a banana’s physical appearance; clarifying that banana is an edible plant; etc). The sentences selected using both START and INTRO appear quite meaningful as well, which is at odds with the relatively poor performance of these methods. The sentences selected using GENERIC seem to be focused on overly specific properties, which may also help to explain the poor performance of this strategy. Although this qualitative analysis suggests that better mentions for each word can be selected using our mention selection strategies, these mention selection strategies no longer pose a clear advantage over random selection as the number of mentions accumulates. On the one hand, we assume that more random mentions of a word make its representation more robust, even if each mention is not informative. On the other hand, each of our proposed mention strategies only focuses on a certain aspect of word meaning and fails to capture more diversified information.

4.5 Summary

We have considered the new challenge of distilling high-quality static word embeddings from language models using only a small number of mentions of each word. Based on our analysis, the most effective strategies are to select sentences using PMI and to include a definition of the target word. The success of these strategies makes it possible to use word embeddings obtained from LMs in applications such as ontology completion and zero shot learning with minimal computational overhead. However, as discussed in the qualitative analysis, our proposed mention selection strategies only focus on a particular aspect of word meaning and fail to capture other aspects. Given this shortcoming, the research of the next chapter aims at obtaining static word embeddings by integrating word representations that capture different aspects of semantic properties.

	banana
RAND	• Born in Puntarenas Province , Lagos ’ parent decided to move to Limón where Cristhian went to school and worked in banana plantation • Binding post or banana plug may be used for lower frequency • In India , vegetarian variety may use potato , calabash , paneer , or banana •A later claim suggested that Bubbles had died ; Jackson’s press agent Lee Solters quipped to the medium that when Bubbles heard about his demise he went banana ... Like Mark Twain , his death is grossly exaggerated and he ’s alive and doing well
HOME	•A banana is an elongated, edible fruit – botanically a berry – produced by several kinds of large herbaceous flowering plants in the genus “Musa" • In some countries, bananas used for cooking may be called “plantains", distinguishing them from dessert bananas • Almost all modern edible seedless (parthenocarp) bananas come from two wild species – “Musa acuminata" and “Musa balbisiana" • They are grown in 135 countries, primarily for their fruit, and to a lesser extent to make fiber, banana wine, and banana beer and as ornamental plants • The scientific names of most cultivated bananas are Musa acuminata, Musa balbisiana, and Musa × paradisiaca for the hybrid Musa acuminata × M. balbisiana, depending on their genomic constitution. The old scientific name for this hybrid, Musa sapientum, is no longer used
INTRO	• The area produces citrus, olives, tomatoes and market-garden vegetables, and is one of the few parts of Europe where commercial banana production is possible • The work, created in an edition of three, consists of a fresh banana taped to a wall with a piece of duct tape • They also sell orange, grape, piña colada, coconut champagne (non-alcoholic), and banana daiquiri (non-alcoholic) fruit drinks • No banana plantation was left unscathed by the hours-long onslaught of strong winds • The crops of highest productivity are plantain, banana, coconut, tomatoes, pepper, eggplant, yucca, rice, beans, maize, "guandules" and sweet potato
PMI	• The common fruits that are used in the preparation include banana, apple, kiwi, strawberry, papaya, pineapple, mango, and soursop • Thus the banana producer and distributor Chiquita produces publicity material for the American market which says that “a plantain is not a banana" • One day Mitchell posted a photo of herself on Twitter next to a bruised banana in response to trolls who had compared her freckles to the overripe fruit • The most important Philippine cooking banana is the saba banana (as well as the very similar cardava banana) • Their meals consist of cooked or steamed rice wrapped in banana or tara or kau leaves that known as “khau how" and boiled vegetables

Table 4.9: Example sentences selected for the word *banana* (1) .

	banana
START	• Bananas, grown mainly for domestic consumption, amount to a steady annual average crop of 70,000 tons. • Bananas were introduced into the americas in the 16th century by portuguese sailors who came across the fruits in west africa, while engaged in commercial ventures and the slave trade" • Bananas must be transported over long distances from the tropics to world markets • Bananas was edited at the time by the now-legendary horror author r. l. stine • Bananas which are turning yellow emit natural ethylene which is characterized by the emission of sweet scented esters
ENUM	• Crops are, for example, cereals (mainly wheat, barley, rye and triticale), soybeans, banana, rice, coffee, turnips, and red as well as sugar beets • These have included: bacon maple ale and chocolate, peanut butter, and banana ale • There are also wild relatives of jackfruit, mango, cardamom, turmeric and banana • Amelita's signature dish was an organic rib fillet with shaved ham, banana, and hollandaise sauce. • Whereas the larger farming plots are utilized for staple crops, families can choose to grow herbs, flowers and fruit trees (mango, banana, plum, orange, lime) in their personal household garden
GENERIC	• Bananas contain more digestible carbohydrates than any other fruit • Bananas have no fat, cholesterol or sodium • Bananas do contain serotonin • Bananas grow on plants • Bananas contain pectin, a soluble fibre
DEF	• Banana is an elongated curved tropical fruit that grows in bunches and has a creamy flesh and a smooth skin

Table 4.10: Example sentences selected for the word *banana* (2).

	falcon
RAND	• This festival focus on Asayel (local camel) and Majahim (dark skinned camel), and also feature falcon hunting, Saluki and Arabian horse race, and date packing contest • In a land where mice eat iron, falcons also kidnap children • While in migration, adult are sometimes preyed on by most of the bird-hunting, larger raptor, especially the peregrine falcon • The falcon "Ida" come to Pkharmat every morning
HOME	• Adult falcons have thin, tapered wings, which enable them to fly at high speed and change direction rapidly • Fledgling falcons, in their first year of flying, have longer flight feathers, which make their configuration more like that of a general-purpose bird such as a broad-wing. • The falcons are the largest genus in the Falconinae subfamily of Falconidae, which itself also includes another subfamily comprising caracaras and a few other specie • The largest falcon is the gyrfalcon at up to 65 cm in length • As with hawks and owls, falcons exhibit sexual dimorphism, with the females typically larger than the males, thus allowing a wider range of prey species
INTRO	• Peregrine falcons, common kestrels and choughs also nest on the cliffs • Many other versions of this song with motif of falcon drinking water from Vardar were published at the beginning of the 20th century in Macedonia (i.e.) • Common birds are: fantails, kingfishers, tui, kereru, New Zealand falcons • It consists of a golden falcon (Hawk of Quraish) with a disk in the middle, which shows the UAE flag and seven stars representing the seven Emirates of the federation • The school mascot is the falcon and the school colors are scarlet and grey
PMI	• The saker falcon is a large hierofalcon, larger than the lanner falcon and almost as large as gyrfalcon at length with a wingspan of • Because he is so often shown with a falcon, he came to be considered the patron saint of falconry • The island has breeding populations of various raptors: golden eagle, buzzard, peregrine falcon, kestrel, hen harrier and short and long-eared owl • The arrangement is intriguing, because normally the Horus falcon and the hieroglyphs inside the serekh were out of reach and independent of one another • Other birds which can be seen include peregrine falcon, merlin, hen harrier, short-eared owl and ring ouzel

Table 4.11: Example sentences selected for the word *falcon* (1) .

	falcon
START	• Falcons of narabedla is a science fiction novel by american writer marion zimmer bradley set in the universe of her darkover series • Falcons rookie lt sam baker was hit in the head in the first half and did not return • Falcons were important in the (formerly often royal) sport of falconry • Falcons defensive end chuck smith questioned the vikings' toughness because of the ease with which they had won during the season • Falcons and cormorants have long been used for hunting and fishing, respectively
ENUM	• The axe did, however, close some country routes including the cuckoo line, the cranleigh line, the steyning line, the new romney branch line and the bexhill west branch line, plus goods yards including deptford wharf and falcon lane • The ford falcon and holden commodore, former chrysler engineers now working for mmal, developed a wider mid-sized car specific to the australian market. • The series features two founding members of the team, ant-man and the wasp, and introduces wonder man, tigra, hawkeye, falcon, vision and scarlet witch • The word perlin is a falconer's term for a cross breed of a peregrine falcon and a merlin • The falcon and the snowman received generally positive notices upon release in 1985 and currently has an 82 percent on rotten tomatoes from 22 critics
GENERIC	• Falcons have long, slim wings which taper to pointed tips • Some falcons eat reptiles • Falcons are small, speedy birds of prey known for their aerial agility • Falcons are birds of prey • Some falcons eat small reptiles
DEF	• Falcon is any bird of the genus Falco, all of which are birds of prey

Table 4.12: Example sentences selected for the word *falcon* (2) .

	bucket
RAND	• A microphone picked up the sound from the bucket , which give it a hollow “megaphone" sound • Another program feature for boy wa the war canoe in which boy went in the lake in two canoe with fire bucket but no paddle • In this maneuver it is possible to lower a bucket on a line to the ground in such a way that the bucket remains stationary on the ground , permitting transfer of material •In a machine of this type , the gravitational force move the drop against the opposing electrostatic field of the bucket • The player only ha a limited supply of slime but slime can be replenished by collecting slime bucket or completing the mystery word
HOME	•A bucket is typically a watertight, vertical cylinder or truncated cone or square, with an open top and a flat bottom, attached to a semicircular carrying handle called the "bail" • A bucket is usually an open-top container • A number of bucket types exist, used for a variety of purposes • Though not always bucket shaped, lunch boxes are sometimes known as lunch pails or a lunch bucket • The bucket has been used in many phrases and idioms in the English language, some of which are regional or specific to the use of English in different English-speaking countries
INTRO	• The pissdale was a 17th-century engineering development: prior to this, crewmen either used buckets or, more frequently, simply urinated over the rails of the ship (though this put them at risk of falling overboard and drowning, as few sailors had any ability to swim) • To clean the sponge, they simply washed it in a bucket with water and salt or vinegar • Linear hashing (LH) is a dynamic data structure which implements a hash table and grows or shrinks one bucket at a time • The woman is carrying a dish in her other hand, and a bucket and a broom have been left in the courtyard • A total of 5,500 bars of soap, 1,600 buckets, 50 baths, and over 180 latrines were distributed to the victims of the devastated area
PMI	• On 17 October 2015 children and parents from the Wiesloch Schiller School () walked to the museum for their annual outing and saw a demonstration of the bucket chain excavator • On Saturday 24 February in 1990 Roman, armed with bucket and secateurs arrived, much to Mr. growers shock when he began picking, himself • Legion of Doom were the first to be eliminated after being disqualified for attacking The Godwinns with their own slop bucket • At approximately 2:40pm an excavator, using a flat scraper bucket to remove the natural clay a few centimetres at a time, struck an object • It’s crowds typically wore baggy clothes and Tie-dye or dayglo colours, with items such as bucket hats, bandanas, dungaree jeans, ponchos and converse sneakers becoming popular

Table 4.13: Example sentences selected for the word *bucket* (1) .

	bucket
START	• Buckets and wells sufficed as a water system for a while, but eventually, the denver water system was created • Buckets can be managed using either the console provided by amazon s3, programmatically using the aws sdk, or with the amazon s3 rest application programming interface (api) • Buckets with plants stand again on the forecourt of the orangery and the extensive areas of herbaceous plants have been restored • Buckets were positioned under leaks to prevent rainwater damaging the main reception rooms • Buckets of shucked oysters were passed through a window into the processing room, where they were rinsed, weighed and tallied
ENUM	• Within that time molly came in with her mop bucket and the colonel and his friends were required to beat a hasty retreat • They would pour water down the trough into a bucket, and the ripe berries would sink into the bucket and the unripe ones would be pushed over the edge to be thrown away. • Steed allows users to transfer files using ftp and sftp protocols and access their bucket and containers on s3 and azure for storing data in the cloud • Fuel was moved by belt bucket and scraper conveyors to the fuel bunkers, then delivered to the boilers by mechanical spreader stokers • Bleacher and bucket seats from vacaville were added ro redding's tiger field during the 2014 renovation that brought the ballpark's capacity to 1,200 seats
GENERIC	• Buckets hanging with several different types of snaps can cause bad eye and nose lacerations • Most buckets fill with water • Bucket wheel current meters are of stainless steel, brass, and bronze construction • Buckets go up and down with the rotation of the shaft • Most buckets have interiors
DEF	• Bucket is a container made of rigid material, often with a handle, used to carry liquids or small items

Table 4.14: Example sentences selected for the word *bucket* (2) .

Chapter 5

Combining Complementary Aspects of Word Meaning

5.1 Introduction

Contextual word representations obtained in different ways may capture different aspects of word meaning. For example, as we have seen in the Chapter 3 and Chapter 4, given a word w and its mention in a sentence s, masking this word and extracting the vector representation of [MASK] can obtain a contextual word embedding. The word embedding obtained in this way is more likely to capture the semantic properties that are satisfied by the corresponding concept. As a result, the neighbours of a given word in the word embedding space are typically its taxonomic siblings. A different approach could be to take the average of the SBERT embeddings of the sentences where a word w appears. This also leads to a representation of w. Still, due to how it has been constructed, we can expect this representation to capture some topical relatedness because we assume that mentions of a target word tend to have some topics.

Essentially, the neighbours of a word in the embedding space are then those words that are likely to belong to the same topic. For some applications, we may need word vectors that capture both semantic properties and topical relatedness, such as few-shot learning and ontology completion. In such cases, the vectors that we have considered in the previous chapters may be too limiting, and underperform standard word embeddings such as skip-gram and GloVe.

Consequently, the research question in this chapter is whether word embeddings that capture both semantic properties and topical relatedness can be obtained by combining different representations. Given two vector representations that capture complementary aspects of the meaning of a single word, the most straightforward way is to combine them with standard aggregation methods such as concatenation or averaging. However, averaging two representations of words suffer from information loss. Although concatenating two word embeddings can keep all the information, it doubles the dimension of embeddings, which can be challenging in downstream applications. Therefore, a related research question in this chapter is how we can combine complementary word embeddings effectively while not increasing their dimension. This chapter aims to answer this research question and analyze to what extent our proposed supervised contrastive learning model can be used to address this issue.

Having been widely applied in computer vision and NLP [39, 51, 58, 57, 55], contrastive learning aims at learning an embedding space in which similar samples are pulled together while diverse ones are pushed far apart [53]. Based on this learning objective, the embedding space learned by contrastive learning models boosts performances on different downstream tasks [39, 51, 58]. These studies suggest that contrastive learning model can capture the inherent general properties from data and generate a high-quality vector space. Furthermore, contrastive learning models can be applied to refine the geometry of a vector space by fixing the anisotropy problem. Anisotropy problem is an issue when the learned embeddings occupy a narrow cone in the vector space, which severely limits their expressive power. [110, 39] confirm that contrastive learning models help to make embeddings evenly distributed in the vector space.

Instead of applying a contrastive learning model on each embedding space individually, we propose a contrastive learning model that can learn from multiple embedding spaces simultaneously and generate a low-dimensional embedding space that captures salient features from all the source embeddings. Motivated by this idea, this chapter aims to integrate complementary word embeddings through a proposed supervised contrastive learning model. Specifically, we aim to answer the following question with analysis.

- Does combining word embeddings that capture different aspects of word meaning lead to word representations that capture both semantic properties and topical relatedness?
- How effective is our proposed contrastive learning model in combining word embeddings?

5.2 Integrating Word Embeddings

In Section 5.2.1, we first describe two strategies for deriving static word vectors and show how these embedding capture different aspects of word meaning. 5.2.3 subsequently describes our proposed contrastive learning model and how it is applied to integrate two word embeddings.

5.2.1 Obtaining Word Embeddings Using Different Methods

To obtain the vector representation of a word w, we first sample n sentences $S1, ..., Sn$ mentioning w. Following the sampling strategy proposed in Chapter 4 (Def+Home), the source corpus from which these sentences are sampled in our experiments is Wikipedia, specifically a dump from March 2021. From each sentence Si, we obtain two vector representations for wi using two following methods:

- **BERT$_{\mathbf{W}}$**: We replace the word w by a [MASK] token and obtain the vector representation of this token in the final layer from the language model BERT.
- **SBERT$_{\mathbf{S}}$**: We take the sentence representation of the mention Si from the language model SBERT as the representation for wi.

In both cases, the final representation of the word w is computed by averaging the vectors that are obtained from a set of sentences.

Regarding the embeddings obtained from $\mathbf{BERT_W}$, we assume that these embeddings capture the taxonomical aspect of word meaning. This is because these vectors correspond to BERT's prediction of the missing word which intuitively captures the taxonomical properties of w. The performance of $MASK$ over $LAST$ in the lexical classification benchmark from Chapter 3 and Chapter 4 also support this assumption. Regarding the embeddings obtained from $\mathbf{SBERT_S}$, we assume that these embeddings capture the topic aspect of word meaning. The reason is that the meanings of different sentences that mention a word w might reveal topics in which w is often involved. We compared the nearest neighbours of random words from two embedding spaces to confirm our assumptions by calculating euclidean distances. As is shown in Table 5.5, the first two columns report the top 10 nearest words measured in vector space of $\mathbf{BERT_W}$ and $\mathbf{SBERT_S}$, which confirm our assumptions. Therefore, we conclude that those two embedding spaces capture different aspects of words meaning. By applying the above two methods, each word $w \in V$ have two embeddings: $\mathbf{w}_{top} \in \mathbb{R}^n$ and $\mathbf{w}_{tax} \in \mathbb{R}^n$.

5.2.2 Identifying Positive Samples

In order to capture salient features from source embeddings, supervised contrastive learning models require both positive and negative samples for each word w. By pulling positive samples toward w while pushing negative samples away from w, contrastive learning models can learn a better embedding space. In applying supervised contrastive learning in computer vision, positive samples are usually generated by flipping or cropping the input image [23, 47]. In NLP, positive samples can also be generated by corrupting the input, such as applying dropout on input vectors [39]. The purpose of generating the positive samples using data augmentation is to force the model to ignore the meaningless variances and to capture the salient features.

Instead of refining the existing source embedding space, the research in this chapter focuses on learning an embedding space that can retain all the salient features from two source embedding spaces without increasing its dimension. We assume that the contrastive learning can capture relevant information by retaining the neighbourhood structure, which share the similar intuition with manifold learning strategies such as [90] and [98]. We propose to use the nearest neighbours for each word w in the two given embedding spaces as its positive samples. Rather than mixing these neighbours from two given embedding space together, our proposed contrastive learning model has two projection layers, and each one only use the positive samples from one source embedding space. The objective of this positive sampling method is to retain the two aspects of meaning captured by the two given source embeddings. Therefore, given word w, we assume that its taxonomically similar words are its nearest neighbours measured in **BERT**$_{\mathbf{W}}$ embedding space. Similarly, we assume that its topically similar words are its nearest neighbours measured in **SBERT**$_{\mathbf{S}}$ embedding space. As a result, w has two sets of positive samples corresponding to taxonomic and topical similarity.

5.2.3 Proposed Contrastive Learning Model

This contrastive learning model aims at retaining both aspects of word meaning from two source embeddings while resulting in a low-dimensional embedding for $w \in V$. Let w be a word from the vocabulary with $\mathbf{w}$ defined as the concatenation of the two source embeddings:

$$\mathsf{w} = \mathsf{w}_{top} \oplus \mathsf{w}_{tax}$$

We learn a Encoder, which is a single linear layer, mapping the embedding from $\mathbf{w} \in \mathbb{R}^n$ to a transformed embedding $\mathbf{v} \in \mathbb{R}^m$, where $m < n$:

$$\mathbf{v} = \mathsf{Encoder}(\mathbf{w})$$

we also learn two linear mappings $Projection_{tax}$ and $Projection_{top}$ from the embedding

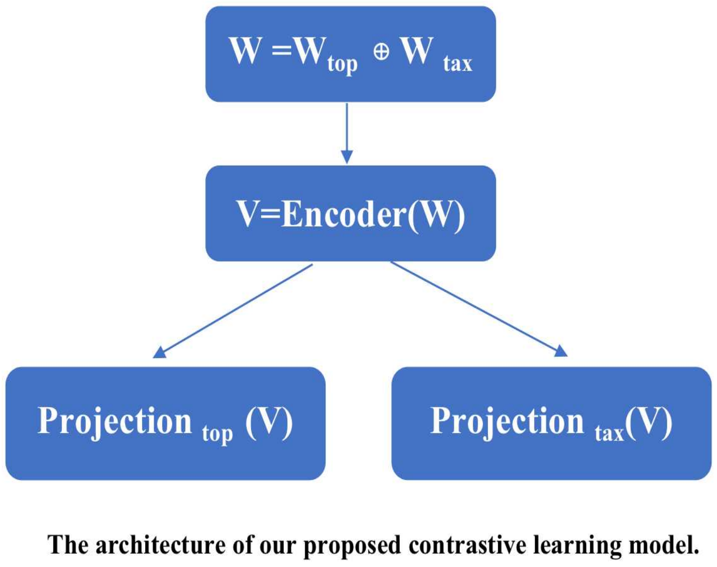

Figure 5.1: The diagram of the proposed contrastive learning model.

$\mathbf{v} \in \mathbb{R}^m$ to transformed embeddings $\mathbf{z}_{tax} \in \mathbb{R}^m$ and $\mathbf{z}_{top} \in \mathbb{R}^m$ corresponding to the structure of embedding spaces from w_{tax} and w_{top}:

$$\mathbf{z}_{tax} = \mathsf{Projection}_{tax}(\mathbf{v})$$

$$\mathbf{z}_{top} = \mathsf{Projection}_{top}(\mathbf{v})$$

The diagram of this contrastive learning model is presented in Figure 5.1. To learn those mappings from the model, we use a loss function that consists of two components:

$$\mathcal{L} = \mathcal{L}_{tax} + \mathcal{L}_{top}$$

where $\mathcal{L}_{tax}$ encourages $\mathbf{v}$ to preserve the information captured by w_{tax}, while $\mathcal{L}_{top}$ inspires $\mathbf{v}$ to preserve the information captured by w_{top}. For the component $\mathcal{L}_{tax}$ we use the supervised contrastive loss following [57]. For each word $w \in V$, $P(w)_{tax}$ is the set of positive examples, which are taxonomically nearest neighbours to w, and $N(w)_{tax}$ is a set of negative examples, which are the words from the current training batch that are not taxonomically similar to w. Note that $P(w)_{tax} \cup N(w)_{tax} \cup \{w\}$ contains every word from the given training batch. Then we define:

$$\mathcal{L}_{tax} = -\sum_{w \in V_{\text{batch}}} \frac{1}{|P(w)_{tax}|} \sum_{w_j \in P(w)_{tax}} \log \frac{\exp(\cos(\mathbf{z}_{tax}, \mathbf{z_j}_{tax})/\tau))}{\sum_{w_l \in N(w)_{tax} \cup P(w)_{tax}} \exp(\cos(\mathbf{z}_{tax}, \mathbf{z_l}_{tax})/\tau))}$$

Likewise, the component $\mathcal{L}_{top}$ is defined in similar way:

$$\mathcal{L}_{top} = -\sum_{w \in V_{\text{batch}}} \frac{1}{|P(w)_{top}|} \sum_{w_j \in P(w)_{top}} \log \frac{\exp(\cos(\mathbf{z}_{top}, \mathbf{z_j}_{top})/\tau))}{\sum_{w_l \in N(w)_{top} \cup P(w)_{top}} \exp(\cos(\mathbf{z}_{top}, \mathbf{z_l}_{top})/\tau))}$$

The temperature value τ in the contrastive loss function was tuned, with possible values chosen from {2, 1, 0.1, 0.07, 0.001}. The learning objective of this proposed contrastive learning model is to retain the taxonomic features and topical feature from its source embedding spaces through minimizing $\mathcal{L}_{tax}$ and $\mathcal{L}_{top}$. We expect the Encoder of this contrastive learning model can capture salient features from two source embedding spaces and generate one embedding space that can integrate the taxonomic and topical aspects of word meaning. To train the model, we repeatedly sample training batches until we obtain all words in the vocabulary V. Each batch contains 1500 words, and each word inside is guaranteed to have at least two positive samples for taxonomic similarity and two positive samples for topical similarity. The training epoch size is set to 30 when its training loss converges.

5.3 Evaluation

We compare the proposed contrastive learning startegy with the individual **BERT**$_{\mathbf{W}}$ and **SBERT**$_{\mathbf{S}}$ strategies themselves, as well as with alternative strategies for combining these embeddings. In Section 5.3.1 we first describe our experimental setup. In Section 5.3.2, we focus on word similarity benchmarks. These benchmarks allow us to assess how well the similarities of different word type can be predicted from the word vectors. This section provides details of datasets and the experiment result analysis. Section 5.3.3 focus on another word embedding evaluation method: lexical classification. In Section 5.3.5, we present a qualitative analysis on the nearest neighbors of different word embeddings.

5.3.1 Experiment Setup

We first consider the two source embeddings obtained from the methods as described in Section5.2.1 and evaluate them individually:

- $\mathbf{BERT_W}$
- $\mathbf{SBERT_S}$

We consider the three possible ways of combining the source embeddings:

- **CONC**: concatenation of the $\mathbf{BERT_W}$ and $\mathbf{SBERT_S}$.
- **AVG**: taking the average of the $\mathbf{BERT_W}$ and $\mathbf{SBERT_S}$.
- **PCA**: applying Principle Component Analysis (PCA) on the **CONC** and reducing its dimension to 768 (same dimension as $\mathbf{BERT_W}$ and $\mathbf{SBERT_S}$).

We consider three method of extracting word embeddings from our contrastive learning model.

- $\mathbf{CT_{encoder}}$: the vector representation from the ENCODER layer of our proposed contrastive learning model.

- **CT$_{\textbf{tax}}$**: the vector representation from the PROJECTION layer corresponding to taxonomic similarity of our proposed contrastive learning model.
- **CT$_{\textbf{topic}}$**: the vector representation from the projection layer corresponding to topic similarity of our proposed contrastive learning model.

We also consider the two standard word embeddings as explained in the Chapter 3:

- **SG**
- **CBOW**

5.3.2 Word Similarity

We use word similarity task as an evaluation method. The purpose of this task is to rank word pairs based on their degree of similarity. This ranking is then compared with a gold standard ranking obtained from human judgements. We consider three word similarity datasets that focus on semantic similarity: the semantic similarity portion of *WordSim*[1], the noun subset of *SimLex999*[49], and *SemEval-17*[21]. These three datasets can help us evaluate the general semantic properties captured by word representations. We also consider the other three word similarity datasets that focus on semantic relatedness: the semantic relatedness portion of *WordSim*[1], the noun subset of *MEN*[18], and *MTurk-771*[45]. Those three datasets can help us evaluate, specifically, the topical aspect of semantic properties captured by word representations. The results of the experiments on these datasets are shown in Table 5.1. First of all, we find **SBERT$_{\text{S}}$** overtake **BERT$_{\text{W}}$** by a substantial margin in five datasets. Among the five datasets, **MEN**, **WordSim-REL**, and **Mturk-771** emphasis on topic aspect of word meaning, which confirm the strength of **SBERT$_{\text{S}}$** in capturing topic features. The underperformance of **BERT$_{\text{W}}$** in **Sim-EVl** and **WordSim-SIM** reveals its shortcoming in representing general semantic properties. Overall, this shows that **SBERT$_{\text{S}}$** captures a different aspect of word meaning which is missing from **BERT$_{\text{W}}$**.

	SimLex-999	Sim-EVl	WordSim-SIM	MEN	WordSim-REL	Mturk-771
BERT$_{W}$	36.8	62.0	49.1	20.6	32.9	46.8
SBERT$_{S}$	36.9	71.4	60.0	30.1	54.1	60.1
CONC	39.7	73.1	62.8	29.6	54.9	62.7
AVG	38.9	73.0	63.9	30.1	55.7	61.9
PCA	38.7	70.8	65.5	30.0	60.8	61.4
CT$_{tax}$	38.9	72.3	51.3	19.4	38.2	56.9
CT$_{topic}$	34.1	70.1	63	28.6	60.2	59.0
CT$_{encoder}$	**42.5**	**74.8**	**67.3**	**30.4**	**62.1**	**65.4**
CBOW	39.2	68.4	60.7	25.3	50.8	62.4
SG	38.6	71.2	67.0	28.3	61.0	**65.4**

Table 5.1: Results for word similarity tasks measured in Spearman correlation. The best results for a given benchmark are shown in bold.

Furthermore, the three methods of combining source embeddings consistently outperform the two source embeddings in five datasets. This result confirms our first hypothesis that **BERT$_{W}$** and **SBERT$_{S}$** capture complementary aspects of word meaning, and combining them gives better word representations. The best results are obtained by **CT$_{encoder}$**, which outperforms all other variants in all six tasks. This support our second hypothesis that our proposed contrastive learning model can effectively capture both aspects of word meaning from two source embeddings in a low-dimensional embedding space. Given that the dimension of **CT$_{encoder}$** is only half of **CONC**, it is a clear benefit for downstream applications that need static word embeddings. The performance of **CT$_{tax}$** and **CT$_{topic}$** are more or less similar with **BERT$_{W}$** and **SBERT$_{S}$**, suggesting that the embeddings from two projection layers of our contrastive learning model resemble the two source embeddings. In addition, the performance of **CBOW** and **SG** shows that those standard word embeddings are still robust and have a clear advantage over contextualized word embedding in word similarity task.

5.3.3 Lexical Classification

Following the previous chapters, we use the four benchmarks and the same evaluation method because those four datasets evaluate word representations based on their capacity to capture different types of semantic properties. As mentioned in the previous chapters, McRae and CSLB focus on commonsense properties, WN supersenses highlight taxonomic properties, whereas BN domains concentrate on topical properties. Furthermore, we add two datasets that focus on concept categorization. The first dataset *BM*, introduced by [8], contains 5321 words divided into 56 categories [1]. The second dataset *AP*, introduced by [4], contains 402 words divided into 21 categories [2]. In our experiments, we have only considered properties/classes for which more than 10 positive examples are available. For both datasets, we used random splits of 60% for training, 20% for tuning and 20% for testing. The datasets are overviewed in Table 5.2.

The results of the experiments on these datasets are shown in Table 5.3. Suprisingly, $\mathbf{SBERT_S}$ slightly outperforms $\mathbf{BERT_W}$ in *McR*, *CSLB*, and *BM*. These results show the capacity and potential of sentence embedding in capturing commonsense semantic properties. We also expect that $\mathbf{SBERT_S}$ can achieve a better result on *BND* as this dataset specifically focus on topic features. However, in contrast to our expectation, $\mathbf{SBERT_S}$ doesn't bring any improvement on *BND*. On the other hand, the experiment results on *WNSS* show that $\mathbf{BERT_W}$ have a clear advantage over $\mathbf{SBERT_S}$. This further confirms the strength of $\mathbf{BERT_W}$ in capturing the taxonomic aspect of word meaning.

Given the complementary strength of $\mathbf{SBERT_S}$ and $\mathbf{BERT_W}$, the experiment results confirm that combining them is beneficial in most benchmarks except for *AP*. However, this benefit is not impressive as its improvements are limited. Furthermore, opposite to

[1]`https://github.com/vecto-ai/word-benchmarks/blob/master/word-categorization/monolingual/en/battig.csv`

[2]`https://github.com/vecto-ai/word-benchmarks/blob/master/word-categorization/monolingual/en/ap.csv`

Dataset	Words	Properties
AP	376	21
BM	2668	55

Table 5.2: Overview of the added two datasets

its outperformance in word similarity task, $\mathbf{CT_{encoder}}$ does not bring a clear improvement over other methods of combining source embeddings. Similar to the result from word similarity benchmarks, both $\mathbf{CT_{tax}}$ and $\mathbf{CT_{tax}}$ underperform $\mathbf{CT_{encoder}}$ in lexical classification, which confirms that the vector representations from the ENCODER layer capture both aspects of meaning from two source embeddings. The performance of **PCA** is still robust, which confirms its effectiveness on dimension reduction, as we found in the Chapter 3.

Overall, the result from lexical classification supports our first hypothesis that $\mathbf{BERT_W}$ and $\mathbf{SBERT_S}$ capture complementary aspects of word meaning and combining them gives better word representations. However, the result also suggests that our proposed contrastive learning model does not bring a clear improvement. Given the outperformance of $\mathbf{CT_{encoder}}$ in word similarity benchmarks, its underperformance in lexical classification might be explained by the linear classifications applied in lexical classification benchmarks. As linear classifiers can extract salient features from input, the well trained linear classifier might cancel the benefit of embedding space learnt by the contrastive learning model.

5.3.4 Lexical substitution

Besides the above two evaluation tasks, we also consider lexical substitution task. This task aims at finding appropriate substitutes for a target word in a sentence. For example, *"great"* in the context *"He is a great artist"* can be substituted by *outstanding*. Lexical substitution is normally used for contextualised language models and can be adapted to evaluate static word embeddings. We consider a specific dataset: ALaSca [61]. This

	McR	CSLB	WNSS	BND	AP	BM
BERT_w	62.0	54.1	59.3	39.1	70.1	72.0
SBERT_s	63.2	56.1	51.7	39.5	69.3	75.4
CONC	64.0	**57.7**	60.9	42.5	68.5	74.4
AVG	**65.3**	56.8	58.2	40.5	67.7	77.1
PCA	63.0	56.3	**62.5**	43.6	66.8	79.2
CT_{tax}	58.4	50.4	54.2	37.3	66.2	67.3
CT_{topic}	60.9	54.6	47.7	38.2	63.8	69.6
$CT_{encoder}$	63.6	56.7	58.5	41.2	70.6	77.2
CBOW	61.1	50.6	48.4	45.0	65.0	72.6
SG	59.6	54.5	55.6	**49.1**	**83.0**	**80.1**

Table 5.3: Results for lexical classification in terms of F1 score. The best results for a given benchmark are shown in bold.

	ALaSca
BERT_sub	0.486
SBERT_sub	0.405
CONC_sub	**0.504**

Table 5.4: Results for lexical substitution. The best results for a given benchmark are shown in bold.

dataset contains 35649 questions, and each question has a sentence mentioning a target word with 10 to 20 substitute options (only one is the correct substitute). For each question, we calculate the cosine similarity between the sentence's representation and its substitutes' representations. These cosine similarity scores are then ranked from large to small. Based on the position of the correct substitution in the rank list, we assign the score 1/rank to it (for example, if the correct substitute is ranked in 3ed place, we give it a score of 1/3). Once we have completed the 35649 questions, the final score is obtained by averaging all the scores.

In this experiment, we consider three different methods to do the task.

- **BERT_sub**: The cosine similarity is computed between the BERT contextual representation of the target word in the sentence and the **BERT_w** representations of its substitute words.
- **SBERT_sub**: The cosine similarity is computed between the SBERT representation of the sentence and the **SBERT_s** representations of its substitute word.
- **CONCA_sub**: In this case, the sentence is represented as the concatenation of the contextual representation of the target word and the SBERT representation of the sentence. Each of its substitute words is represented by the concatenation of **BERT_w** and **SBERT_s**.

The experimental result shows that **CONCA_sub** outperforms both **BERT_sub** and **SBERT_sub**, suggesting that BERT and SBERT are complementary and combining them is beneficial in lexical substitution task, which further support our research assumption.

5.3.5 Qualitative Analysis

In Table 5.5, 5.6, and 5.7, we show the top 10 nearest neighbours of ten randomly selected words in five embedding spaces: $\mathbf{BERT_W}$, $\mathbf{SBERT_S}$, $\mathbf{CT_{encoder}}$,$\mathbf{CT_{tax}}$, and $\mathbf{CT_{topic}}$. Some of the listed examples clearly illustrate the difference between $\mathbf{BERT_W}$ and $\mathbf{SBERT_S}$. For example, in the case of *spring*, the top 10 nearest neighbours from $\mathbf{BERT_W}$ include all other seasons, such as winter and fall, whereas the top 10 nearest neighbours from $\mathbf{SBERT_S}$ are words topically related to *spring*, such as *sandwort*, *spirea*, and *bloom*. This result supports our hypothesis that $\mathbf{BERT_W}$ captures the taxonomic aspect of word meaning while $\mathbf{SBERT_S}$ captures the topic aspect of word meaning. The nearest neighbours from $\mathbf{CT_{encoder}}$ reveal that our proposed contrastive learning model successfully captures both aspects of word meaning. For instance, in the case of *school*, its nearest neighbors from $\mathbf{CT_{encoder}}$ contain both topically similar words such as *education*, *student* and taxonomic similar words like *kindergarten*, *college*. Moreover, rather than simply retaining the geometries of its source embeddings, $\mathbf{CT_{encoder}}$ even generate a better neighbourhood structure by reconciling the two source embeddings. For example, the top three nearest neighbours of *spring* in $\mathbf{CT_{encoder}}$ are *mid-autumn*, *late-spring*, and *mid-summer*, which are semantically closer to *spring* than the corresponding nearest neighbours in $\mathbf{BERT_W}$ and $\mathbf{SBERT_S}$. Surprisingly, *fall*, *winter*, and *sandwort* are not included in the top ten near neighbours of *spring* even though they are fed as positive examples to the model during training. Likewise, we also found that the $\mathbf{CT_{encoder}}$ sometimes favours one aspect of word meaning over the other. For example, In the case of *job*, the top 15 nearest neighbours from $\mathbf{CT_{encoder}}$ do not include taxonomic neighbours *handout* which is fed as the positive examples during training. Since the contrastive learning model integrates two aspects of word meaning by minimizing two loss functions simultaneously during the training, it might ignore some positive examples to avoid the increased overall loss and thus generate a better neighbourhood structure.

The embedding spaces of $\mathbf{CT_{tax}}$ and $\mathbf{CT_{topic}}$ are more or less similar to the embedding spaces of $\mathbf{BERT_W}$ and $\mathbf{SBERT_S}$, which are their corresponding learning objectives during the training. For example, the top three nearest neighbour of *school* in $\mathbf{BERT_W}$ are *university*,*institution*, and *curriculum*, whereas its counterparts in $\mathbf{CT_{tax}}$ are *curriculum*, *university*, and *college*. Likewise, the top three nearest neighbour of *school* in $\mathbf{SBERT_S}$ are *education*,*student*, and *college*, whereas its counterparts in $\mathbf{CT_{top}}$ are *student*, *education*, and *college*. However, like $\mathbf{CT_{encoder}}$, $\mathbf{CT_{tax}}$ and $\mathbf{CT_{topic}}$ also have their own neighbourhood structure that is not the same as its learning target. For example, the second nearest neighbour of *job* in $\mathbf{CT_{tax}}$ is *chore*, which is not found in the top ten nearest neighbours in $\mathbf{BERT_W}$. Likewise, the second nearest neighbour of *job* in $\mathbf{BERT_W}$ is *handout*, which is ranked below *chore* by $\mathbf{CT_{tax}}$. This example suggests that $\mathbf{CT_{tax}}$ makes *chore* closer to *job* than *handout*, even though *handout* is fed as a positive example during the training.

5.4 Summary

Based on the experiment results, we found that MASK vectors that we have been studying in the previous two chapters only capture one aspect of word meaning, and that they can be improved by also modelling topical similarity. In this chapter, we have proposed to use SBERT embeddings to capture this topical similarity and we found that combining SBERT embeddings and BERT-MASK is beneficial. We also have proposed a strategy for combining word embeddings through a contrastive learning model that generate low-dimensional embeddings. For both word similarity and word classification, we found that combining the MASK and SBERT vectors leads to improved results. For word similarity, our contrastive strategy was also consistently better than other combination strategies, although this benefit was not seen for word classification.

BERT$_W$	**SBERT$_S$**	**CT$_{encoder}$**	**CT$_{tax}$**	**CT$_{topic}$**
		Top 10 Nearest Neighbors of *textbook*		
casebook	coauthored	handbook	casebook	handbook
handbook	handbook	monograph	monograph	two-volume
monograph	authoritative	treatise	journal	learning
workbook	instructive	book	handbook	seminal
tome	learning	encyclopedia	encyclopaedia	thesis
encyclopaedia	book	essay	workbook	pedagogical
preprints	class	encyclopaedia	tome	authoritative
bibliography	scholarly	two-volume	preprints	chrestomathy
book	seminal	thesis	brochure	bibliography
brochure	specialisation	bibliography	encyclopedia	multi-volume
		Top 10 Nearest Neighbors of *school*		
university	education	education	curriculum	student
institution	student	student	university	education
curriculum	college	curriculum	college	college
lyceum	pre-primary	lyceum	institute	coeducational
education	coeducational	kindergarten	department	lyceum
institute	state-funded	college	classroom	educational
preschool	lyceum	university	specialism	kindergarten
training	curriculum	madrassahs	academy	boys-only
kindergarten	non-faith	pedagogy	kindergarten	institute
academy	university	pre-school	class	curriculum
		Top 10 Nearest Neighbors of *job*		
profession	employment	career	position	employment
handout	career	employment	chore	self-employed
training	work-related	workplace	paperwork	career
position	employable	profession	career	apprenticeship
hobby	apprenticeship	training	handout	work-related
sinecure	work-related	apprenticeship	wait-list	workplace
rota	menial	work-related	care	receptionist
traineeship	workforce	freelancer	checkbook	caseworker
drudgery	unemployment	trainee	training	tutoring
career	workplace	nine-to-five	homework	internship

Table 5.5: Top 10 nearest neighbors of selected words(1) .

BERT$_W$	**SBERT$_S$**	**CT$_{encoder}$**	**CT$_{tax}$**	**CT$_{topic}$**
		Top 10 Nearest Neighbors of *map*		
schematic	mapmaking	mapmaking	atlas	mapmaking
document	mapmakers	atlas	schematic	atlas
printout	atlas	topography	gazetteer	triangulation
horoscope	cartography	datum	cryptogram	cartography
photograph	topography	mapmakers	diagram	datum
atlas	raised-relief	cartography	flowchart	geo
bookmark	datum	globe	abacus	topography
microdot	surveying	geo	snapshot	heightmap
image	geo	surveying	marker	surveying
gazetteer	north	planisphere	facsimile	chart
		Top 10 Nearest Neighbors of *spring*		
fall	sandwort	mid-autumn	fall	early-flowering
mid-summer	mayflower	late-spring	winter	meadowsweet
winter	spirea	mid-summer	mid-winter	avens
summertime	harebell	dandelion	mid-summer	spirea
summer	mayapple	blooming	mid-autumn	mayflower
mid-autumn	bloom	flowered	summertime	goldilocks
mid-winter	meadowsweet	bloom	late-spring	bloom
autumn	goldthread	wildflower	mid-afternoon	rosebay
late-spring	dogwood	lilly	late-fall	daffodil
late-fall	globeflower	milkweed	autumn	twinflower
		Top 10 Nearest Neighbors of *red*		
green	color	yellow-red	green	yellow
yellow	trichromatic	crimson	yellow	purple
yellow-red	yellow-red	yellow	white	magenta
sky-blue	colour	purple	blue	yellow-red
blue	hue	violet	black	hue
maroon	crimson	magenta	yellow-red	colour
white	red-orange	red-orange	light-blue	violet
light-blue	all-red	all-red	sky-blue	cyan
crimson	magenta	purplish-red	pink	red-orange
red-white	brownish-red	colour	aquamarine	blue

Table 5.6: Top 10 nearest neighbors of selected words(2) .

BERT$_W$	**SBERT$_S$**	**CT$_{encoder}$**	**CT$_{tax}$**	**CT$_{topic}$**
		Top 10 Nearest Neighbors of *math*		
civics	eighth-graders	precalculus	civics	eighth-graders
numeracy	precalculus	mathematics	study-abroad	college-level
mathematics	measurer	college-level	cosmetology	student
phonics	college-level	numeracy	undergraduate	one-semester
coursework	mathematics	high-school	stenography	precalculus
writing	aptitude	eighth-graders	numeracy	student-faculty
science	summative	coursework	librarianship	measurer
elocution	matriculants	reading	economics	numeracy
reading	kindergartner	measurer	science	sub-test
stenograph	decile	aptitude	librarianship	high-school
		Top 10 Nearest Neighbors of *small*		
tiny	smallish	tiny	tiny	sized
modest-sized	modest-sized	modest-sized	subterranean	tiny
subterranean	tiny	smallish	modest-sized	modest-sized
rudimentary	sized	moderate-sized	large	nestle
large	small-scale	sized	serpentine	include
hollow	embayment	smaller	smaller	beneath
broad	flat-topped	small-scale	circular	one-inch
circular	smaller	flat-topped	rudimentary	cairn
smaller	valley-like	similar-sized	large-sized	atop
remote	vicinity	smaller-scale	remote	sited
		Top 10 Nearest Neighbors of *animal*		
ruminant	rearing	livestock	ruminant	wildlife
invertebrate	slaughtering	wild-animal	invertebrate	zookeepers
vertebrate	wild-animal	bovid	anuran	wild-animal
primate	hoofed	zookeepers	vertebrate	road-killed
anuran	livestock	buffalo	human	undomesticated
shellfish	domesticate	camelid	shellfish	feral
mammal	zookeepers	piglet	amphibian	buffalo
homeotherm	meat-eating	wildlife	mammal	domesticate
earthworm	aurochs	camelids	amphibia	rearing
amphibian	graze	semi-wild	homeotherm	domestic

Table 5.7: Top 10 nearest neighbors of selected words(3) .

Chapter 6

Conclusions and Future Work

This final chapter provides a summary of the research conducted in the thesis. First, it relates the contributions to the thesis hypothesis and summarises the main findings. Subsequently, we address each of the considered research questions. It ends by showing some possible directions for future work.

6.1 Thesis Summary and Contributions

The primary motivation for this thesis is learning the representation of words. The prevalence of pre-trained language models gives rise to contextualised word embeddings, which represent words at the token level. However, the meaning of a word at the type level is missing, which hinders the applications where word meaning has to be modelled in the absence of context, such as ontology completion and few-shot learning. In order to learn word representation at the type level and take advantage of the rich knowledge preserved in contextualised language models, this thesis aims to distil static word embeddings from contextualised language models. The research hypothesis for this thesis was presented in Chapter 1 as follows: better static word embeddings can be efficiently distilled from CLMs by strategically selecting sentences and combining complementary methods. We have developed several methods, as presented in Chapter 3, Chapter 4, and Chapter 5, to test this hypothesis.

After summarising the related work in Chapter 2, Chapter 3 proposed using topic models to improve how word mentions are sampled. The topic model used in this research is Latent Dirichlet Allocation. First, given the collected contexts of each word, Latent Dirichlet Allocation has been applied to partition all the mentions into several clusters corresponding to different topics. Then, topic-specific vectors for each word are obtained. Finally, these topic-specific vectors are evaluated on the lexical classification task, which tests how well various semantic properties can be predicted from the word vectors. Based on the performances of vectors obtained by different mentioning strategies, we found that selecting an equal number of mentions per topic outperforms purely random selection strategies. This result supports the idea that better static word embeddings can be distilled from CLMs by selecting sentences using the topic model.

In Chapter 4, we considered the new challenge of distilling high-quality static word embeddings from CLMs using only a small number of mentions of each word. We aim to solve this challenge by analysing a range of strategies for selecting a few mentions of a given the word w. Then, we compare these strategies by evaluating their generated word vectors on lexical classification and ontology completion tasks. Based on the analysis of the experiment results, we found that using Pointwise Mutual Information (PMI) and definition sentences lead to better static word vectors. These word vectors outperform the vectors obtained from 500 randomly selected mentions on the four lexical classification benchmarks. This research supports that high-quality word embedding can be obtained from just a few highly-informative mentions of each word.

In Chapter 5, we found that MASK vectors that we have been studying in the Chapter 3 and Chapter 4 only capture the taxonomic aspect of word meaning and that they can be further improved by adding the word embeddings that capture the topical aspect of word meaning. In this chapter, we proposed using SBERT embeddings to capture this topical aspect of word meaning, and we found that combining SBERT embeddings and BERT-MASK is beneficial. We also have proposed a strategy for combining word embeddings through a contrastive learning model that generates low-dimensional embeddings. Moreover, the low-dimensional embeddings learnt from this contrastive learning model can retain the relevant neighbourhood structure of both SBERT vectors and MASK vectors. Thus, they preserve two aspects of semantic meaning captured by SBERT vectors and MASK vectors. We found that combining the MASK and SBERT vectors improves results for both word similarity and classification tasks. Our contrastive strategy was consistently better than other combination strategies in all 6 word similarity benchmarks.

To conclude, the experimental results obtained from Chapter 3, Chapter 4, and Chapter 5 show that the quality of word embeddings distilled from CLMs are improved by applying the methods we proposed. This result supports our research hypothesis that better static word embeddings can be efficiently distilled from CLMs by strategically selecting sentences and combining complementary methods.

6.2 Research Questions

In this section, the research questions previously identified in Chapter 1 will be revisited and discussed regarding the relation between each question and the research conducted in this thesis.

- **Research Question 1:** *Can higher-quality word embeddings be obtained by selecting sentences strategically?*

  To answer this question, we adopted different strategies to select sentences and evaluated the quality of the word vectors obtained by those strategies. We selected sentences using topic models in Chapter 3 and other strategies in Chapter 4. The evaluation results from both chapters suggest that strategic sentence selection leads to more informative sentences, generating better word embeddings.

- **Research Question 2:** *Can higher-quality word embeddings be obtained from a few mentions of each word?*

To answer this question, we restricted the number of mentions to 20 at maximum and explored a range of sentence selection strategies. The finding from Chapter 4 shows that 20 mentions of each word obtained using PMI and definition sentences lead to word vectors that outperform word vectors obtained by 500 random mentions. This result suggests that higher-quality word embeddings can be obtained by just a few mentions of each word with the help of strategic selections.

- **Research Question 3:** *Can higher-quality word embeddings be obtained by combining representations that capture complementary aspects of word meaning?*

  The research from Chapter 5 found that MASK vectors are good at capturing the taxonomic aspect of word meaning. In contrast, SBERT vectors exhibits expertise in capturing the topic aspect of word meaning. These two methods of obtaining word embeddings are complementary. The evaluation result supports that combining MASK and SBERT improves word embeddings.

6.3 Future Work

This section discusses possible approaches to extend the research presented in this thesis for potential future work.

- **Obtaining Word Embeddings by Fine-tuning CLMs:**

  The strategies proposed in this research to distil static word embeddings from CLMs without fine-turning or prompting CLMs are far from optimum. This is because CLMs are trained to do tasks such as next-word-prediction, which prioritise the parameters setting that might be less optimal for distilling word embeddings. On the other hand, fine-tuning CLMs on tasks such as lexical classification can encourage CLMs to shift their focus to model the semantic properties of words. Likewise, this also can be achieved by prompting CLMs.Therefore, distilling word vectors by fine-turning or prompting CLMs can be the next step to continue the research in this thesis.

- **Mention Generation by Large Language Models**

  Although the mention selection strategies proposed in this thesis can be applied to obtain more informative mentions, their effectiveness depend on the quantity and the quality of the available corpus from which the mentions are sampled. Given the limited computation power, increasing the size of corpus to select the mentions can be time consuming and thus inefficient. On the other hand, large language models are generally trained in massive corpora, capable of generating high-quality sentences. For example, ChatGPT, a large language generation model released by OpenAI company, have impressive capacity to engage in human-like dialogue based on a prompt. This makes large language models excellent options for obtaining informative mentions efficiently. Specifically, we can collecting high quality mentions by prompting large langugae models, such as asking: "can you please give me more information about w". Therefore, generating mentions from large language models can be more efficient and effective in obtaining informative sentences and produce even better word representations.

- **Improving Word Representations for Under-Resourced Languages**

  Acquiring large amounts of data is essential to build NLP applications. It is not an issue for most representative languages, such as English, Chinese, and Spanish, as they have large language communities that generate data day by day. However, there are over 6000 languages used in this world [6], and many of them are under-resourced. Therefore, it is not feasible to build language models such as BERT or even word embeddings such as Word2Vec for languages that don't have enough data. As mentioned in the previous chapters, one application of static word embeddings is zero-shot or few-shot learning. Suppose we can align the word vector space of English and an under-resourced language by learning a matrix. In that case, we can improve the vector representations of words in an under-resourced language and facilitate its NLP application.

Bibliography

[1] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Pasca, and A. Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. 2009.

[2] I. Alghanmi, L. E. Anke, and S. Schockaert. Combining bert with static word embeddings for categorizing social media. In *Proceedings of the Sixth Workshop on Noisy User-generated Text*, pages 28–33, 2020.

[3] F. Almeida and G. Xexéo. Word embeddings: A survey. *arXiv preprint arXiv:1901.09069*, 2019.

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

[5] A. Amrami and Y. Goldberg. Towards better substitution-based word sense induction. *arXiv preprint arXiv:1905.12598*, 2019.

[6] S. R. Anderson. How many languages are there in the world. *Linguistic Society of America*, 2010.

[7] M. Apidianaki. From word types to tokens and back: A survey of approaches to word meaning representation and interpretation. *Computational Linguistics*, pages 1–60, 2022.

[8] M. Baroni, B. Murphy, E. Barbu, and M. Poesio. Strudel: A corpus-based semantic model based on properties and types. *Cognitive science*, 34(2):222–254, 2010.

[9] M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, 2014.

[10] Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

[11] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[12] S. Bhakthavatsalam, C. Anastasiades, and P. Clark. GenericsKB: A knowledge base of generic statements. *CoRR*, abs/2005.00660, 2020.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[15] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[16] R. Bommasani, K. Davis, and C. Cardie. Interpreting pretrained contextualized representations via reductions to static embeddings. In *Proceedings ACL*, pages 4758–4781, 2020.

[17] R. Bommasani, K. Davis, and C. Cardie. Interpreting pretrained contextualized representations via reductions to static embeddings. In *Proceedings of the 58th*

Annual Meeting of the Association for Computational Linguistics, pages 4758–4781, 2020.

[18] E. Bruni, N.-K. Tran, and M. Baroni. Multimodal distributional semantics. *J. Artif. Intell. Res.*, 49(1-47), 2014.

[19] J. Camacho-Collados and R. Navigli. BabelDomains: Large-scale domain labeling of lexical resources. In *Proceedings EACL*, pages 223–228, 2017.

[20] J. Camacho-Collados, M. T. Pilehvar, and R. Navigli. NASARI: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1059. URL `https://aclanthology.org/N15-1059`.

[21] J. Camacho-Collados, M. T. Pilehvar, N. Collier, and R. Navigli. 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, 2017.

[22] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

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

[24] G. Chronis and K. Erk. When is a bishop not like a rook? when it's like a rabbi! multi-prototype bert embeddings for estimating semantic relationships. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 227–244, 2020.

[25] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*, 2019.

[26] A. Conneau and D. Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

[27] J. Dan and H. Martin James. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2009.

[28] J. Davison, J. Feldman, and A. M. Rush. 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, 2019.

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

[30] P. Dufter, N. Kassner, and H. Schütze. Static embeddings as efficient knowledge bases? *arXiv preprint arXiv:2104.07094*, 2021.

[31] K. Ethayarajh. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings EMNLP*, pages 55–65, 2019.

[32] K. Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[33] A. Ettinger. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48, 2020.

[34] M. Faruqui and C. Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, 2014.

[35] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*, 2016.

[36] J. R. Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

[37] M. Forbes, A. Holtzman, and Y. Choi. Do neural language representations learn physical commonsense? *arXiv preprint arXiv:1908.02899*, 2019.

[38] L. Gan, Z. Teng, Y. Zhang, L. Zhu, F. Wu, and Y. Yang. Semglove: Semantic co-occurrences for glove from bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:2696–2704, 2022.

[39] T. Gao, X. Yao, and D. Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[40] P. Gardenfors. *The geometry of meaning: Semantics based on conceptual spaces*. MIT press, 2014.

[41] Y. Goldberg. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*, 2019.

[42] Y. Gong, H. Luo, and J. Zhang. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*, 2017.

[43] B. Guo, C. Zhang, J. Liu, and X. Ma. Improving text classification with weighted word embeddings via a multi-channel textcnn model. *Neurocomputing*, 363:366–374, 2019.

[44] P. Gupta and M. Jaggi. Obtaining better static word embeddings using contextual embedding models. *arXiv preprint arXiv:2106.04302*, 2021.

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

[46] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[47] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[48] J. Hewitt and C. D. Manning. 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, Volume 1 (Long and Short Papers)*, pages 4129–4138, 2019.

[49] F. Hill, R. Reichart, and A. Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.

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

[51] D. T. Hoffmann, N. Behrmann, J. Gall, T. Brox, and M. Noroozi. Ranking info noise contrastive estimation: Boosting contrastive learning via ranked positives. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 897–905, 2022.

[52] A. Imani, A. Vakili, A. Montazer, and A. Shakery. Deep neural networks for query expansion using word embeddings. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part II 41*, pages 203–210. Springer, 2019.

[53] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

[54] P. N. Johnson-Laird. The mental representation of the meaning of words. *Cognition*, 25(1-2):189–211, 1987.

[55] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus. Hard negative mixing for contrastive learning. *Advances in Neural Information Processing Systems*, 33:21798–21809, 2020.

[56] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha. Ammus: A survey of transformer-based pretrained models in natural language processing. *arXiv preprint arXiv:2108.05542*, 2021.

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

[58] T. Kim, K. M. Yoo, and S.-g. Lee. Self-guided contrastive learning for bert sentence representations. *arXiv preprint arXiv:2106.07345*, 2021.

[59] P. Kolyvakis, A. Kalousis, and D. Kiritsis. Deepalignment: Unsupervised ontology matching with refined word vectors. In *Proceedings NAACL-HLT*, pages 787–798, 2018.

[60] O. Kwon, D. Kim, S.-R. Lee, J. Choi, and S. Lee. Handling out-of-vocabulary problem in hangeul word embeddings. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3213–3221, 2021.

[61] C. Lacerra, T. Pasini, R. Tripodi, R. Navigli, et al. Alasca: an automated approach for large-scale lexical substitution. In *IJCAI*, pages 3836–3842, 2021.

[62] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL `https://aclanthology.org/D17-1082`.

[63] O. Levy and Y. Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-2050. URL `https://aclanthology.org/P14-2050`.

[64] A. Li, W. Huang, X. Lan, J. Feng, Z. Li, and L. Wang. Boosting few-shot learning with adaptive margin loss. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12573–12581, 2020.

[65] N. Li, Z. Bouraoui, and S. Schockaert. Ontology completion using graph convolutional networks. In *Proceedings of the 18th International Semantic Web Conference*, pages 435–452, 2019.

[66] N. Li, Z. Bouraoui, and S. Schockaert. Ontology completion using graph convolutional networks. In *Proceedings ISWC*, pages 435–452, 2019.

[67] N. Li, Z. Bouraoui, J. C. Collados, L. Espinosa-Anke, Q. Gu, and S. Schockaert. Modelling general properties of nouns by selectively averaging contextualised embeddings. *arXiv preprint arXiv:2012.07580*, 2020.

[68] Q. Liu, D. McCarthy, and A. Korhonen. Towards better context-aware lexical semantics: Adjusting contextualized representations through static anchors. In *Proceedings EMNLP*, pages 4066–4075, 2020.

[69] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pre-training approach. *CoRR*, abs/1907.11692, 2019.

[70] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[71] H. Luo, Z. Liu, H. Luan, and M. Sun. Online learning of interpretable word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1687–1692, 2015.

[72] K. McRae et al. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37:547–559, 2005.

[73] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings ICLR*, 2013.

[74] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[75] G. A. Miller. Wordnet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[76] B. Murphy, P. Talukdar, and T. Mitchell. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING 2012*, pages 1933–1950, 2012.

[77] M. L. Murphy. *Lexical meaning*. Cambridge University Press, 2010.

[78] R. Navigli and P. Velardi. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1318–1327, 2010.

[79] K. D. Onal, Y. Zhang, I. S. Altingovde, M. M. Rahman, P. Karagoz, A. Braylan, B. Dang, H.-L. Chang, H. Kim, Q. McNamara, et al. Neural information retrieval: At the end of the early years. *Information Retrieval Journal*, 21(2-3): 111–182, 2018.

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

[81] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`.

[82] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

[83] N. Poerner, U. Waltinger, and H. Schütze. E-bert: Efficient-yet-effective entity embeddings for bert. *arXiv preprint arXiv:1911.03681*, 2019.

[84] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

[85] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[86] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[87] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://aclanthology.org/D16-1264`.

[88] E. Reif, A. Yuan, M. Wattenberg, F. B. Viégas, A. Coenen, A. Pearce, and B. Kim. Visualizing and measuring the geometry of BERT. In *Proceedings NeurIPS*, pages 8592–8600, 2019.

[89] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings EMNLP*, pages 3982–3992, 2019.

[90] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[91] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[92] L. K. Şenel, F. Şahinuç, V. Yücesoy, H. Schütze, T. Çukur, and A. Koç. Learning interpretable word embeddings via bidirectional alignment of dimensions with semantic concepts. *Information Processing & Management*, 59(3):102925, 2022.

[93] V. Shwartz and Y. Choi. Do neural language models overcome reporting bias? In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6863–6870, 2020.

[94] R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng. Zero-shot learning through cross-modal transfer. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 935–943, 2013.

[95] R. Speer, J. Chin, and C. Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[96] R. A. Stein, P. A. Jaques, and J. F. Valiati. An analysis of hierarchical text classification using word embeddings. *Information Sciences*, 471:216–232, 2019.

[97] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A large ontology from wikipedia and wordnet. *Journal of Web Semantics*, 6(3):203–217, 2008.

[98] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[99] I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

[100] I. Tenney, P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. Van Durme, S. R. Bowman, D. Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019.

[101] E. Thiéblin, O. Haemmerlé, N. Hernandez, and C. Trojahn. Survey on complex ontology matching. *Semantic Web*, 11(4):689–727, 2020.

[102] L. Thompson and D. Mimno. Topic modeling with contextualized word representation clusters. *arXiv preprint arXiv:2010.12626*, 2020.

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

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

[105] I. Vulic, E. M. Ponti, R. Litschko, G. Glavas, and A. Korhonen. Probing pre-trained language models for lexical semantics. In *Proceedings EMNLP*, pages 7222–7240, 2020.

[106] I. Vulić, E. M. Ponti, R. Litschko, G. Glavaš, and A. Korhonen. Probing pretrained language models for lexical semantics. *arXiv preprint arXiv:2010.05731*, 2020.

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

[108] B. Wang, A. Wang, F. Chen, Y. Wang, and C.-C. J. Kuo. Evaluating word embedding models: methods and experimental results. *APSIPA transactions on signal and information processing*, 8, 2019.

[109] S. Wang and J. Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015.

[110] T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

[111] W. Wang, V. W. Zheng, H. Yu, and C. Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019.

[112] Y. Wang, Z. Bouraoui, L. E. Anke, and S. Schockaert. Deriving word vectors from contextualized language models using topic-aware mention selection. *arXiv preprint arXiv:2106.07947*, 2021.

[113] Y. Wang, L. Cui, and Y. Zhang. Improving skip-gram embeddings using bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1318–1328, 2021.

[114] Y. Wang, Z. Bouraoui, L. Espinosa-Anke, and S. Schockaert. Sentence selection strategies for distilling word embeddings from bert. 2022.

[115] N. Weir, A. Poliak, and B. Van Durme. Probing neural language models for human tacit assumptions. *arXiv preprint arXiv:2004.04877*, 2020.

[116] F. Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*, pages 196–202. Springer, 1992.

[117] C. Xing, N. Rostamzadeh, B. N. Oreshkin, and P. O. Pinheiro. Adaptive cross-modal few-shot learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 4848–4858, 2019.

[118] K. Yan, Z. Bouraoui, P. Wang, S. Jameel, and S. Schockaert. Aligning visual prototypes with BERT embeddings for few-shot learning. In *Proceedings of the International Conference on Multimedia Retrieval*, pages 367–375, 2021.

[119] L. Yao, C. Mao, and Y. Luo. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*, 2019.

[120] Y. Zhang, X. Wang, S. Lai, S. He, K. Liu, J. Zhao, and X. Lv. Ontology matching with word embeddings. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data: 13th China National Conference, CCL 2014, and Second International Symposium, NLP-NABD 2014, Wuhan, China, October 18-19, 2014. Proceedings*, pages 34–45. Springer, 2014.

[121] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*, 2019.