Analyzing the Effect of Cross-lingual Transfer, Compression and Whitening on Natural Language Encoding

自然言語符号化における
言語横断的な転移学習、圧縮、白色化の効果の分析

Shota Sasaki

Graduate School of Information Sciences
Tohoku University

This dissertation is submitted for the degree of
Doctor of Information Science

August 2022

Acknowledgements

本研究を進めるにあたり，多くの方々のご協力，ご助言を頂きましたことに感謝申し上げます．主指導教員である乾健太郎教授には，研究はもちろんのこと，キャリアや私生活に関することまで，親身に相談に乗って頂きました．また理化学研究所に勤めながら，社会人博士学生として研究を行うという大変貴重な機会を設けてくださいました．理化学研究所においては，教育関連事業者様との共同研究にも参加させて頂き，規模・影響力が非常に大きいプロジェクトの推進を任せて頂きました．私のこれからの人生の礎となる経験をさせて頂き，誠に感謝いたします．

篠原歩教授，大町真一郎教授，ならびに鈴木潤教授は，ご多忙の最中，審査委員として本論文を査読してくださいましたこと，深く感謝いたします．鈴木潤教授には，修士課程の頃から，共著者として多くの研究をサポートして頂きました．時には xx 時間に渡る長時間の議論を交わしてくださったこと，感謝いたします．RIKEN AIP の Benjamin Heinzerling さん，奈良先端大の大内啓樹助教，北陸先端大の井之上直也准教授，東京工業大の高瀬翔助教には，メンターとして研究の指針や，実装レベルの課題，そしてマインドの持ち方についてまで，多くのアドバイスを頂きましたこと，心から感謝いたします．研究者としてのロールモデルである皆様との関わりを通じて，私の研究の，また研究者としての確かな核を形成することができました．

Johns Hopkins 大学の Kevin Duh 先生は，私が修士学生の時に，私を研究インターン生として快く受け入れてくださりました．そこでの研究を通して，研究の本当の楽しさ知ることができました．Kevin 先生との出会いがなければ，私が博士課程に進学することはなかったと思います．私の研究者としての道を開いてくださったこと，誠に感謝いたします．留学先での生活を支えてくださった，当時 John Hopkins 大学に在籍なさった坂口慶祐准教授，そして JHU CLSP のメンバー，特にルームメイトであった Jinyi Yang さん，Samik Sadhu さん，Xiaofei Wang さん，そしてホームステイ先の Valerie Long さんに感謝いたします．

研究の相談から日常の雑談まで，様々な面で関わってくださった株式会社サイバーエージェントの三田雅人さん，Apple Inc. の吉川将司さん，RIKEN AIP の Ana Brassard さん，塙一晃さん，LINE 株式会社の清野舜さん，ヤフー株式会社の浅野

広樹さんに感謝いたします．研究室の事務員としていつもサポートしてくださった菅原真由美さん，相澤知佳さん，そして RIKEN AIP での共同研究を支えてくださった磯部順子さん，AIP アシスタントの皆さんに感謝いたします．研究室配属時から研究室の先輩として多くのことを教えてくださった松林優一郎准教授，山口健史さん，代勤さん，LINE 株式会社の水本智也さんに感謝いたします．

共同研究を行った後輩の舟山弘晃さん，アクセンチュア株式会社の佐藤俊さんに感謝いたします．烈々とした勢いで成長する姿から刺激を受け，私自身の成長にも繋げることができました．また，サーバーマネージャーの仕事や，外部の共同研究，プライベートで関わることの多かった伊藤拓海さん，阿部香央莉さん，小林悟郎さんに感謝いたします．

研究をしている中で，時には辛くなる時もありました．そんな時に，お酒を酌み交わし，研究での大変なことを忘れさせてくれた山形東高校サッカー部の同期部員，東北大学サッカー部の同期部員に感謝いたします．最後に，これまで見守ってくれた家族に感謝いたします．

Abstract

A *text encoder* is an essential component of almost all deep Natural Language Processing (NLP) models. Its role is to convert the input text into embeddings, i.e., vectors that represent the meaning of the text. As a result of various refinements of the text encoder, deep NLP models now perform incredibly well, however, only on a limited number of tasks. In addition, recent models lack practical applicability as they are not controllable due to being too large.

This dissertation focuses on further improving text encoders from two perspectives: *quality* and *applicability*. A high-quality text encoder must provide suitable encodings that transform the information obtained from the text without any loss of information. A text encoder must also be widely applicable in the real world where language has ever-evolving usages and forms. Aiming for high-quality and widely applicable text encoders, we explore three aspects with potential for improvement: (i) the required amount of training data, (ii) memory requirements for embeddings and unknown words, and (iii) biases in embeddings.

We first propose a cross-lingual transfer learning method to address the data size problem and demonstrate that it improves model performance for low-resource languages. Specifically, as an example of an application of NLP, we conducted experiments on information retrieval tasks. We also clarify the model architecture for information retrieval in which the transfer learning works.

We then propose a method that combines memory sharing and key-value-query (KVQ) operation idea in order to simultaneously reduce the memory requirement for static word embeddings and deal with the unknown word problem. The proposed method succeeded in reducing the amount of required memory while acquiring unknown word embeddings which performed well on word similarity tasks.

Finally, we examine the semantic effects of *whitening*, one of the isotropization methods for anisotropic embeddings, i.e., reducing spatial bias. From our experiments on a sentence similarity task, we report that the effect on static word embeddings has significant overlap with the effect of removing word frequency bias, and that this is not the case for contextualized embeddings.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

With the recent revival of deep learning, many conventional models have been replaced by deep learning models in Natural Language Processing (NLP). A *text encoder* is an essential component of almost all deep NLP models, from text classification models to text generation models (e.g., translation models). The role of a text encoder is to convert the input text into embeddings, i.e., vectors that represent the meaning of the text. At the embedding layer—the entrance of a text encoder—static word embeddings, such as Word2Vec (Mikolov et al., 2013b) and GloVe (Pennington et al., 2014a), transform discrete information, i.e., word sequences, into continuous values that machines can handle. The model then further transforms these values through task-specific layers until they reach a final output layer designed to produce values appropriate for the task at hand. The advent of word embeddings has dramatically increased the affinity between NLP and deep learning and has played a leading role in introducing deep learning into the field of NLP. Since then, methods of obtaining word, phrase, and sentence embeddings using deeper neural networks have also emerged. For example, deep *contextualized* embeddings, such as BERT (Devlin et al., 2019), are the most widely used word embeddings in recent years.

As a result of these various refinements of the text encoder, deep NLP models perform incredibly well. In some tasks, they even reportedly exceeded human correctness rates (Kiela et al., 2021). A question then naturally arises—*Have NLP models already reached their goal?* The answer is "No." Satisfactory performance has only been achieved in a limited number of tasks, and still, challenging tasks such as discourse comprehension and common sense reasoning tasks remain. In addition, recently proposed models lack practical applicability in that they are not controllable due to being too large or their black-box nature.

This dissertation focuses on further improving text encoders from two perspectives: *quality* and *applicability*. A high-quality text encoder must provide suitable encodings that transform the information obtained from the text without losing any information. Input texts are typically very diverse, consisting of combinations of millions of words; a text encoder must generalize and learn the usage and meaning of the language, usually using large training data.

A text encoder must also be widely applicable. Language is a living organism with ever-evolving usages and forms. Therefore, text encoders need to be flexible enough to adapt. In the real world, different speakers use different languages with various language styles. In addition, new words that have not been used before will appear. It also needs to work with compact devices, such as mobile devices or IoT devices, such as smartphones and home assistants, which do not benefit from the vast computational capacities of research or development environments. Establishing an encoder that satisfies these requirements is essential for future real-world application of the NLP models. The ultimate goal of this line of research is to establish a *universal text encoder* with "perfect quality" and "wide applicability."

1.1 Research Issues

This dissertation moves toward the goal of high-quality and widely applicable text encoders by exploring three aspects with potential for improvement: (i) the required amount of training data, (ii) memory requirements for embeddings and unknown words, and (iii) biases in embeddings.

The data size problem: can low-resource languages attain high-resource-level performance? Deep learning models are notoriously data-hungry, meaning that a large amount of data is needed to train the model; the same is true for text encoders. A handful of languages may have adequate data, but many more do not have that privilege, limiting the performance of NLP models using text encoders in those languages. Waiting for equally vast amounts of data to become available for each language to match high-resource-level performance is impractical. Thus, an alternative solution is needed to improve the quality of text encoders in cases where simply increasing the amount of data is not possible.

Memory requirements vs. vocabulary coverage. The strongest text encoders are also usually very large—unrealistically so for the capacity of devices such as mobile phones or IoT appliances which require their use. A naive approach to reducing word embeddings' memory requirement is limiting their vocabulary coverage. However, an unknown word (OOV; out-of-vocabulary) problem becomes critical, which reduces their usefulness in real-world applications where diverse vocabularies are expected. There is a trade-off between reducing the required amount of memory and the capacity to deal with unknown words. Thus, a balanced approach that reduces the memory requirement while maintaining the ability to handle OOV words would improve the applicability of text encoders.

Bias in embeddings. Various undesired biases in word embeddings can negatively impact a model's task performance. Even leaving the impact on performance aside, biased models can be problematic for real-world applications. For example, dialogue generation models with a gender bias have been reported to produce gender discriminatory statements (Nicholson, 2022). Thus, biases in word embeddings pose a barrier to real-world applications in both quality and applicability. Various methods have been proposed to mitigate undesired biases. One of them, *whitening*, has attracted attention due to being simple yet effective in improving performance and mitigating spatial bias. However, while it is clear that whitening mitigates spatial bias in word embeddings, its *semantic* effects (e.g., removal of word frequency bias) are not clear.

1.2 Contributions

This dissertation makes the following contributions:

Investigation of the effectiveness of cross-lingual transfer learning. We propose a cross-lingual transfer learning method and demonstrate that it improves model performance for low-resource languages. Specifically, as an example of an application of NLP, we conducted experiments on information retrieval tasks. We also clarify the model architecture for information retrieval in which the transfer learning works.

Reducing memory requirements and handling unknown words. In order to simultaneously reduce the memory requirement for static word embeddings and deal with the unknown word problem, we propose a method that combines memory sharing and key-value-query (KVQ) operation idea. The proposed method succeeded in reducing the amount of required memory while acquiring unknown word embeddings which performed well on word similarity tasks.

Examining the effect of whitening on word embeddings. We examined the semantic effects of *whitening*, one of the isotropization methods for anisotropic embeddings, i.e., reducing spatial bias. From our experiments on a sentence similarity task, we report that the effect on static word embeddings has significant overlap with the effect of removing word frequency bias, and that this is not the case for contextualized embeddings.

1.3 Overview

The rest of this dissertation is structured as follows:

Chapter 2: Background. This chapter describes the components of text encoders, their characteristics, and their evolution and provides the background on current issues.

Chapter 3: Cross-lingual Transfer Learning for Information Retrieval. This chapter presents our examination of the effectiveness of cross-lingual transfer learning in information retrieval tasks. Specifically, we propose a method in which a task-specific embedding layer that has already been trained on high-resource language data is used to initialize models for a low-resource language.

Chapter 4: Subword-based Compact Reconstruction for Open-vocabulary Word Embeddings. In this chapter, we demonstrate simultaneously reducing the amount of memory required for static word embeddings and address the unknown word problem. We propose a method that combines memory sharing and KVQ operation idea in a subword-based framework, and verify its effectiveness.

Chapter 5: Examining the Effect of Whitening on Static and Contextualized Word Embeddings. In this chapter, we examine the semantic effects of the whitening transformation, which is known to remove bias in word embeddings. We apply a reconstruction-based frequency bias removal method and whitening simultaneously to word embeddings and observe the behavior of the model to confirm the relationship between their effects.

Chapter 6: Conclusion. We summarize the contributions of this dissertation.

Chapter 2

Background

The main theme of this dissertation is to improve a text encoder that is an essential component of almost all deep NLP models. This chapter explains the internal components of a text encoder and their evolution.

2.1 Text Encoder

A text encoder consists of two main components: (i) an embedding layer and (ii) a task-specific layer. An embedding layer is a general component that models for various tasks can share. Its role is to serve as an entry point of a text encoder, converting input texts into embeddings (vectors) that machines can handle. On the other hand, a task-specific layer has a different network architecture for each task and basically cannot be shared across tasks. A task-specific layer acts as a bridge between an embedding layer and the goal of a target task. The resultant vector from a text encoder is used to determine an output of a model. For example, if the model is a classifier for a text classification task, the encoded vector and the softmax function are used to calculate the probability of each target class. If the model is a translation model, the encoded vector is passed to a decoder, which outputs a translation result. We note that, in this study, we refer to not only the encoder in an encoder-decoder model (Sutskever et al., 2014) but also the encoding component in broader kinds of models, e.g., text classification models, as a text encoder.

2.1.1 Embedding Layer

While a task-specific layer needs to perform a task-specific transformation, such as a language mapping in a translation task, an embedding layer, as the entry point to a text encoder, needs to transform input texts into embeddings that machines can handle. To achieve this, an embedding layer must capture languages' universal properties and usages. Generally, these requirements are satisfied through learning using a large unlabeled text corpus such as web crawling data and Wikipedia dumps. This learning process is similar to the process of humans acquiring the usage of languages by hearing many languages over a long period from their birth. In the following paragraphs, we explain the evolution of an embedding layer from static word embeddings to contextualized embeddings.

Static Word Embeddings "Word" has long been recognized as the most intuitive and smallest unit for constructing the meaning of a sentence. At the beginning of the deep learning era of NLP, *distributional representations* were used as the semantic representations of words. Distributional representations are low-dimensional embeddings of words obtained through dimensionality reduction of a word co-occurrence matrix. This process is based on distributional hypothesis (Harris, 1954); words' meanings are determined by the words in their contexts. Following distributional representations, word embeddings from neural language models (Bengio et al., 2003) have received more attention. Word embeddings are low-dimensional vectors assigned to words in advance, then updated through the learning of a language model task. Skip-gram and CBOW (Mikolov et al., 2013a), also known as Word2Vec, received a great deal of attention for their success in acquiring representations that capture relationships between words (word analogies). With the advent of these word embeddings, the affinity between NLP and deep learning has increased dramatically, and they have played a leading role in introducing the trend of deep learning into the field of NLP.

Despite the success of static word embeddings, several drawbacks have been pointed out due to their static nature. The methods of static word embeddings basically assign a vector to each word in a predefined vocabulary. The assigned vector retains the meaning of the word itself without considering its contexts. This property becomes problematic when the scope of the target is expanded to "sentence". A naive method to obtain sentence representations is averaging embeddings of words in a sentence. However, such a method can not model the property of languages that the meaning of words varies depending on their contexts.

Towards wider scope: to embed the meaning of words in contexts Embedding methods with an extended scope to contexts of words have also been explored. Skip-thoughts (Kiros et al., 2015) is a method based on static word embeddings and recurrent neural networks (RNN), learning general embeddings of sentences. With the evolution of a GPU, models with more layers and larger hidden states have been explored. ELMo (Peters et al., 2018b) and BERT (Devlin et al., 2019) successfully learned word embeddings with contextualized property, i.e., meaning changes appropriately according to contexts. GPT-3 (Brown et al., 2020) showed its human-like document generation ability and drew a great deal of attention in the NLP field.

While performance improvements were reported, undesired biases in embeddings caused by using a large raw corpus as training data have became apparent. It was reported that a dialogue generation model made gender discriminatory utterances (Nicholson, 2022). In addition, the huge internal parameter set and the fact that many high-performance GPUs is required for the use of the models have raised doubts about their applicability in the real world.

2.1.2 Task-specific Layer

Following the establishment of an embedding layer, a task-specific layer has been explored in various ways on a task-by-task basis. A task-specific layer has a different network architecture for each task, serving as a bridge between an embedding layer and a target task goal. Following the success of word embeddings, architectures that compose word embeddings to obtain phrase, sentence, and document embeddings have been explored. A convolutional neural network (CNN) has been used in text classification models as a task-specific layer to compose word embeddings (Kim, 2014). RNN, long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), and gated recurrent unit (GRU) (Cho et al., 2014) incorporated into a sequence-to-sequence model have had great success in translation tasks. These networks were established as a general architecture that views a sentence as a word sequence rather than a bag-of-words (BoW), and have been applied to a wide range of tasks.

The data used to train an embedding layer is unlabeled text data, which is relatively easy to obtain in large quantities. In contrast, regarding task-specific data, data shortage problems tend to occur. Specifically data belonging to special categories such as minor domains or minor languages is difficult to collect in large scale. One method addressing this problem is *transfer learning*, which exploits data from other domains or languages different from a target one.

Chapter 3

Cross-lingual Transfer Learning for Information Retrieval

3.1 Introduction

Multilingual document collections are becoming prevalent. Thus an important application is cross-lingual information retrieval (CLIR), i.e. document retrieval which assumes that the language of the user's query does not match that of the documents. For example, imagine an investor who wishes to monitor consumer sentiment of an international brand in Twitter conversations around the world. She might issue a query string in English, and desire all relevant tweets in any language.

There are two main approaches to building CLIR systems. The *modular approach* involves a pipeline of two components: translation (machine translation or bilingual dictionary look-up) and monolingual information retrieval (IR). These approaches may be further divided into the *document translation* and *query translation* approaches (Nie, 2010). In the former, one translates all foreign-language documents to the language of the user query prior to IR indexing; in the latter, one indexes foreign-language documents and translates the query. In both, the idea is to solve the translation problem separately, so that CLIR becomes document retrieval in the monolingual setting.

A distinctly different way to build CLIR systems is what may be called the *direct modeling approach* (Bai et al., 2010; Sokolov et al., 2013). This assumes the availability of CLIR training examples of the form (q, d, r), where q is an English query, d is a foreign-language document, a r is the corresponding relevance judgment for d with respect to q. One directly builds a retrieval model $S(q, d)$ that scores the query-document pair. While q and d are in different languages, the model directly learns both translation and retrieval relevance on the

CLIR training data. Compared to the modular approach, direct modeling is advantageous in that it focuses on learning translations that are beneficial for retrieval, rather than translations that preserve sentence meaning/structure in bitext.

However, there exist no large-scale CLIR dataset that can support direct modeling approaches in a wide variety of languages. To obtain relevance judgments, one typically needs a bilingual speaker who can read a foreign-language document and assess whether it is relevant for a given English query. This can be an expensive process. Here, we present a large-scale dataset that is automatically constructed from Wikipedia: it can support training and evaluation of CLIR systems between English queries and documents in 25 other languages (Section 3.2). The data is of sufficient size for direct modeling, and can also serve as an wide-coverage evaluation data for the modular approaches.[1]

To demonstrate the utility of the data, we further present experiments for CLIR in low-resource languages. First, we introduce a neural CLIR model based on the direct modeling approach (Section 3.3.1). We then show how we can bootstrap CLIR models for languages with less training data by an appropriate use of paramater sharing among different language pairs (Section 3.3.2). For example, using the training data for Japanese-English CLIR, we can improve the Mean Average Precision (MAP) results of a Swahili-English CLIR system by 5-7 points (Section 3.4).

3.2 Large-scale CLIR Dataset

We construct a large-scale CLIR data from Wikipedia. The idea is to exploit *inter-language links*: from an English page, we extract a sentence as query, and label the linked foreign-document pages as relevant. See Figure 3.1 for an illustration.

This data construction process is similar to (Schamoni et al., 2014) who made an English-German CLIR dataset, but ours is at a larger scale. Specifically, we use Wikipedia dumps released on August 23, 2017. English queries are obtained by extracting the first sentence of every English Wikipedia article. The intuition is that the first sentence is usually a well-defined summary of its corresponding article and should be thematically related for articles linked to it from another language. Similar to (Schamoni et al., 2014), title words from the query sentences are removed, because they may be present across different language editions. This deletion prevents the task from becoming an easy keyword matching task.

For practical purposes, each document is limited to the first 200 words of the article. Empty documents and category pages are filtered. Currently, our dataset consists of more

[1]To facilitate CLIR research, the dataset is publicly available at http://www.cs.jhu.edu/~kevinduh/a/wikiclir2018/.

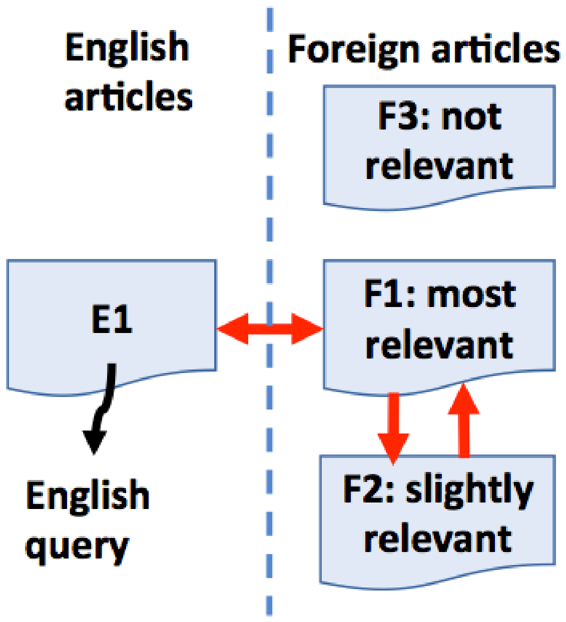


Figure 3.1 CLIR data construction process: From an English article (E1), we extract the English query. Using the inter-language link, we obtain the *most relevant* foreign-language document (F1). Any article that has mutual links to and from F1 are labeled as *slightly relevant* (F2). All other articles are *not relevant* (F3). The data is a set of tuples: (English query q, foreign document d, relevance judgment r), where $r \in \{0, 1, 2\}$ represents the three levels of relevance.

than 2.8 million English queries and relevant documents from 25 other selected languages (see Table 3.1).

In sum, we have created a CLIR dataset that is large-scale in terms of both the amount of examples as well as the number of languages. This can be used in two scenarios: (1) one mixed-language collection where an English query may retrieve relevant documents in multiple languages. (2) 25 independent datasets for training and evaluating CLIR on English queries against one foreign language collection. In the experiments in Section 3.4, we will utilize the dataset in terms of scenario (2).[2]

[2]For extensibility purposes, these experiments use only half of the data, randomly sampled by query (the held-out data is reserved for other uses). Also it only considers binary relevance (*most relevant* vs *not relevant*) for simplicity. The exact data splits will be provided along with the data release.

3.3 Direct Modeling for CLIR

3.3.1 Neural Ranking Model

Given an English query q and a foreign-language document d, our models compute the relevance score $S(q, d)$. First, we represent each word as n-dimensional vectors, so q and d are represented as matrices $\mathbf{Q} \in \mathbb{R}^{n \times |q|}$ and $\mathbf{D} \in \mathbb{R}^{n \times |d|}$, where $|q|$ and $|d|$ are the numbers of tokens in q and d:

$$\begin{aligned} \mathbf{Q} &= [E_q(q_1); E_q(q_2); ...; E_q(q_{|q|})] \\ \mathbf{D} &= [E_d(d_1); E_d(d_2); ...; E_d(d_{|d|})] \end{aligned}$$

q_i and d_i denote the i-th term in q and d. E is embedding function which transforms each term to a dense n-dimensional vector as its representation. ; is the concatenation operator. Then, we apply convolutional feature map[3] to these matrices, followed by tanh activation and average-pooling to obtain each representation vector $\hat{q}$ and $\hat{d}$.

$$\hat{q} = CNN_q(\mathbf{Q}); \quad \hat{d} = CNN_d(\mathbf{D}) \tag{3.1}$$

Next, we define two variations in calculating $S(q, d)$. The first is a *cosine model* which computes cosine similarity between $\hat{q}$ and $\hat{d}$:

$$S_{cos}(q, d) = cossim(\hat{q}, \hat{d}) \tag{3.2}$$

The second is a *deep model* with a fully connected layer on top of the concatenation of $\hat{q}$ and $\hat{d}$ (a 200-dimensional vector):

$$\begin{aligned} S_{deep}(q, d) &= tanh(O \cdot h_{vec}^{\mathrm{T}}) \\ &= tanh(O \cdot relu(W \cdot [\hat{q}; \hat{d}]^{\mathrm{T}})) \end{aligned} \tag{3.3}$$

Here, $O \in \mathbb{R}^{1 \times h}$ and $W \in \mathbb{R}^{h \times 200}$ are the deep model parameters, and h is the number of dimensions of the hidden state, $h_{vec} \in \mathbb{R}^{1 \times h}$. For regularization, we set dropout rate as 0.5 (Srivastava et al., 2014) at the hidden layer.

In the training phase, we minimize pairwise ranking loss, which is widely used for learning-to-rank (Dehghani et al., 2017; Guo et al., 2016; Hui et al., 2017; Pang et al., 2016;

[3]The $n \times 4$ convolution window has filter size of 100 and takes a stride of 1.

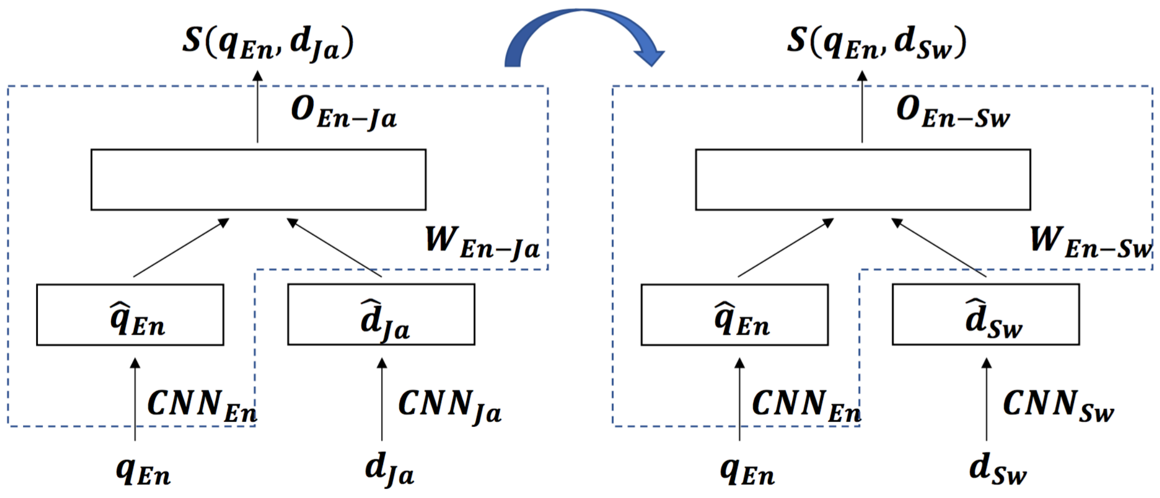


Figure 3.2 Illustration of the proposed method. On low resource dataset (e.g. Swahili-English), the parameters of the CNN for encoding query (CNN_{En}) and the parameters of the fully connected layer (O_{En-Sw}, W_{En-Sw}) are initialized by the ones pre-trained on high resource dataset (e.g. Japanese-English).

Xiong et al., 2017), defined as follows:

$$L = max\left\{0, 1 - (S(q, d^{+}) - S(q, d^{-}))\right\} \tag{3.4}$$

where d^+ and d^- are relevant and non-relevant document respectively. We fix only the word embeddings and tune the other parameters.

We note there are many other ranking models that can be adapted to CLIR (Huang et al., 2013; Mitra et al., 2017; Shen et al., 2014; Xiong et al., 2017); they have a common framework in extracting features from both query and document and optimizing scores $S(q, d)$ via some ranking loss.

Language	**#Doc**	**#Query**	**#SR**
Arabic	535	324	194
Catalan	548	339	625
Chinese	951	463	462
Czech	386	233	720
Dutch	1908	687	1646
Finnish	418	273	665
French	1894	1089	4048
German	2091	938	4612
Italian	1347	808	2635
Japanese	1071	426	2912
Korean	394	224	343
Norwegian-Nynorsk	133	99	150
Norwegian-Bokmål	471	299	663
Polish	1234	693	1777
Portuguese	973	611	1130
Romanian	376	199	251
Russian	1413	664	1656
Simple English	127	114	135
Spanish	1302	781	2113
Swahili	37	22	35
Swedish	3785	639	1430
Tagalog	79	48	23
Turkish	295	185	195
Ukrainian	704	348	565
Vietnamese	1392	354	257
(All numbers are in units of one thousand)			

Table 3.1 CLIR dataset statistics. For each language X, we show the total number of documents in language X and the number of English queries. The number of ”most relevant” documents is by definition equal to #Query. The number of ”slightly relevant” documents is shown in the column #SR.

	Ja	De	Fr
$S_{cos}(q, d)$: cos	59/74	49/66	55/70
$S_{deep}(q, d)$: h=100	61/75	64/77	69/81
$S_{deep}(q, d)$: h=200	68/80	67/79	74/84
$S_{deep}(q, d)$: h=300	70/82	70/81	74/84
$S_{deep}(q, d)$: h=400	**73**/83	**71/82**	75/**85**
$S_{deep}(q, d)$: h=500	**73/84**	70/81	**76/85**

Table 3.2 P@1/MAP performance (0-100 range, in percent) of the cosine model and the deep model with different hidden state size on **high resource datasets**. Best value in each column is highlighted in bold.

	Tl			Sw			De (subsample)			Fr (subsample)		
	In	Sh	Δ	In	Sh	Δ	In	Sh	Δ	In	Sh	Δ
cos	51/68	50/67	-/-	51/67	49/65	-/-	40/59	38/56	-/-	46/63	43/60	-/-
h=100	34/50	48/62	+/+	46/62	46/62	=/=	39/55	46/62	+/+	40/57	46/62	+/+
h=200	44/58	55/67	+/+	47/63	52/67	+/+	41/57	48/63	+/+	43/60	51/66	+/+
h=300	42/57	49/63	+/+	50/65	58/70	+/+	44/60	50/65	+/+	49/65	51/66	+/+
h=400	49/63	**57/69**	+/+	51/66	**60/73**	+/+	45/61	**51/66**	+/+	47/64	**56/70**	+/+
h=500	51/64	54/67	+/+	53/68	56/69	+/+	44/60	49/65	+/+	47/63	51/66	+/+

Table 3.3 P@1/MAP performances on **low resource datasets**. Δ columns show the comparison between the basic deep models with in-language training (In) and the deep models with sharing parameters (Sh); + indicates Sh outperforms In, and - indicates the In outperforms Sh. Best value in each dataset is highlighted in bold.

3.3.2 Sharing Representations

Training a network like the deep model generally requires a nontrivial amount of data. To address the data requirement for low-resource languages, we propose a simple yet effective method that shares representations across CLIR models trained in different language-pairs. Basically, we use the same architecture as the deep model ($S_{deep}(q, d)$, Equation 3.3). However, we use the parameters trained on a high-resource dataset (e.g Japanese-English) to initialize the parameters for a low-resource language-pair (e.g. Swahili-English).

Figure 3.2 illustrates the idea: Concretely, we initialize the parameters of the CNN for encoding query (CNN_q) and the parameters of the fully connected layer (O, W) by using the pre-trained parameters. When training on low-resource data, we fix only the word embedding, and tune the parameters of CNNs and the fully connected layer.

The intuition behind this is that our direct modeling approach enforces $\hat{q}$ and $\hat{d}$ to become language-independent representations of the query and document. The parameters O and W in the deep layer can therefore be used for any language-pair. Note for the cosine model, we can also share parameters for CNN_q.

3.4 Experiment Results

Setup: We use datasets of 3 high-resource languages (Japanese [Ja], German [De], French [Fr]) and 2 low-resource languages (Tagalog [Tl], Swahili [Sw]). We also subsample German and French data to be equivalent to the size of Swahili, in order to compare training size effects. Word embedding with dimension 100 for each language is trained on Wikipedia corpus, using word2vec SGNS (Mikolov et al., 2013b). The size of hidden states in the deep model is {100, 200, 300, 400, 500}. We adopt Adam (Kingma and Ba, 2014) for optimization, train for 20 epochs and pick the best epoch based on development set loss. For the proposed method of parameter sharing, we use the weight parameters pre-trained on Japanese-English dataset to initialize parameters.

High-resource results: Table 3.2 shows the P@1 (precision at top position) and MAP (mean average precision) for datasets consisting of on the order of 100k+ training queries. The deep models outperformed the cosine models under all conditions, suggesting that the fully connected layer can exploit the large training set in learning more expressive scoring functions.

Low-resource results: Table 3.3 shows the results on the low resource datasets under two conditions: training on only the language-pair of interest (in-language), or additionally sharing parameters using a pre-trained Japanese-English model. For the in-language case, we observe the cosine model outperforms the deep model. In contrast to the high-resource results, this implies that deep models, which have a lot of parameters, only become effective if provided with sufficient training data.

For the sharing case, the deep models with parameter sharing outperformed the basic deep models trained only on in-language data under almost all conditions. This indicates that our sharing method reduces training data requirement. Importantly, by sharing parameters, the deep models are now able to outperform the cosine model and achieve the best results on all datasets.[4]

3.5 Conclusion and Future Work

We introduce a large-scale CLIR dataset in 25 languages. This enables the training and evaluation of direct modeling approaches in CLIR. We also present a neural ranking model with shared representations, and demonstrate its effectiveness in bootstrapping CLIR in low-resource languages.

Future work includes: (a) expansion of the dataset to more languages, (b) extraction of different types of queries and relevant judgments from Wikipedia, and (c) development of other ranking models. Importantly, we also plan to evaluate our models on standard CLIR test sets such as TREC (Schäuble and Sheridan, 1997), NTCIR (2007), FIRE (2013) and CLEF (2016). This will help answer the question of whether knowledge learned from automatically-generated datasets can be transferred to a wide range of CLIR problems.

[4]Sharing representations with the cosine models did not help; we hypothesize that cross-lingual sharing only works if given sufficient model expressiveness. We also tried the shared deep models on high resource datasets (e.g. using Japanese parameters on the full French dataset without subsampling). As expected, results did not change significantly.

Chapter 4

Subword-based Compact Reconstruction for Open-vocabulary Word Embeddings

4.1 Introduction

Machine-readable representation of word meanings is one of the essential tools for tackling natural language understanding by computers. A recent trend is to embed word meanings into a vector space by using the rapidly developing neural word embedding methods, such as Skip-gram (Mikolov et al., 2013c), GloVe (Pennington et al., 2014a), and fastText (Bojanowski et al., 2017). The basic idea used to construct a vector space model is derived from the intuition that similar words tend to appear in similar contexts (Miller and Charles, 1991). These methods have successfully been proven to capture high-quality syntactic and semantic relationships in a vector space. For example, studies in compositional semantics have revealed that the calculations underlying embedding vectors, such as addition and inner product, can be considered satisfactory approximations of the composed word meaning and the similarity between words, respectively (Mikolov et al., 2013c). Furthermore, pre-trained word embeddings, especially those trained on a vast amount of text data, such as the Common Crawl (CC) corpus[1], are now considered as highly beneficial, fundamental lan-

[1]http://commoncrawl.org

guage resources for tackling many applications in the AI field. Typical examples of large, well-trained word embeddings are those trained on the CC corpus with 600 billion tokens by fastText (Bojanowski et al., 2017) and with 840 billion tokens by GloVe (Pennington et al., 2014a), which we refer to as `fastText.600B`[2] and `GloVe.840B`[3], respectively. Such word embeddings are often used to improve performance in many natural language processing (NLP) tasks, such as constituency parsing (Gómez-Rodríguez and Vilares, 2018; Suzuki et al., 2018), discourse parsing (Yu et al., 2018), semantic parsing (Dong and Lapata, 2018; Groschwitz et al., 2018), and semantic role labeling (Strubell et al., 2018).

However, well-trained word embeddings have several limitations, and in this paper, we focus on two issues surrounding them: (i) the massive memory requirement and (ii) the inapplicability of out-of-vocabulary (OOV) words. Addressing such issues is crucial, especially when applying them to real-world open systems. The total number of embeddings (i.e., the total memory requirement of such word embeddings) often becomes unacceptably large, especially in limited-memory environments, including GPUs, because the vocabulary size is more than 2 million words, requiring at least 2 gigabytes (GB) of memory for storage.

The memory requirement could be straightforwardly reduced by merely discarding the less important words from the vocabulary. However, such a naive method worsens the drawback of the inapplicability of OOV words, whose handling is highly desirable in real systems because input words can be uncontrollably diverse. Therefore, at present, there is a trade-off between the number of embedding vectors and the applicability of OOV words. Our goal is to investigate and develop a method that simultaneously equips *less memory requirement* and *high applicability of OOV words*, both desirable properties for word embeddings in real-world open systems.

A popular approach for mitigating or solving OOV word issues is to use subword information. Conceptually, the subword-based approach covers all words that can be constructed by a combination of subwords. Thus, the subword-based approach can greatly mitigate (or solve) the OOV word issue. In this study, we extend this approach to simultaneously reduce the total number of embedding vectors through the reconstruction of word embeddings by using subwords. The key techniques of our approach are twofold: memory-shared embeddings and a variant of the key-value-query (KVQ) self-attention mechanism (Vaswani et al., 2017). That is, our approach reconstructs well-trained word embeddings by using a limited number of embedding vectors that are shared by all the subwords with an effective weighting calculated by the self-attention mechanism.

[2]https://fasttext.cc/docs/en/english-vectors.html
[3]https://nlp.stanford.edu/projects/glove/

We experimentally show that our reconstructed subword-based embeddings can successfully imitate well-trained word embeddings, such as `fastText.600B` and `GloVe.840B`, in a small fixed space while preventing quality degradation across several linguistic benchmark datasets from word similarity and analogy tasks. We also demonstrate the effectiveness of our reconstructed embeddings for representing the embeddings of OOV words. Finally, we confirm the performance of our reconstructed embeddings in several downstream tasks such as the named entity recognition task and the textual entailment task.

Our prior conference paper (Sasaki et al., 2019) introduced the basic idea of the proposed method. The basic idea is to characterize subwords by their subword indexes given by hash functions. The indexes are restricted to a fixed size bucket so that models can have reduced memory requirements. We extend this method that handles multiple hash functions as it allows us to perform the multi-mapping of subword embedding vectors, to further improve the performance (Section 4.4.2). Moreover, as the primary extension of our paper, we increase the reliability of our analyses and conclusions by conducting extensive, comprehensive experiments:

(i) model shrinkage experiments on `GloVe.840B` embeddings as the reconstruction target (Figs. 4.5 and 4.6 and Tables 4.5 and 4.6 in Section 4.6.1),

(ii) synthetic OOV experiments on `GloVe.840B` embeddings as the reconstruction target embeddings (Table 4.7 in Section 4.6.2),

(iii) synthetic OOV experiments on analogy datasets (Table 4.7 in Section 4.6.2),

(iv) calculation speed analysis including an investigation of the influence of the multi-mapping process (Section 4.7.1),

(v) more in-depth analyses of the model behaviors on the semantic and syntactic analogy tests (Fig. 4.7 in Section 4.7.2), and

(vi) comprehensive evaluations to investigate the impacts of hyper-parameters, such as F in the combination methods and the scaling hyper-parameter Z in a softmax function (Figs. 4.8 and 4.9 in Section 4.7.3), and

(vii) investigation of the distribution of `FNV` hash function (Fig. 4.11 in Section 4.7.4).

These additional experiments strongly support the usefulness of the proposed method.

The remainder of the paper is organized as follows. First, in Section 4.2, we introduce the studies related to our proposed method. Next, we provide the definition of baseline conventional neural word embeddings and describe the problem setting of our target task in Section 4.3. Then, we present the proposed method in Section 4.4. We explain the three distinct settings of our experiments, (i) evaluation of the model shrinkage performance on standard semantic meaning representations, (ii) evaluation of the performance on OOV words, and (iii) evaluation of the performance of applying word embeddings to downstream tasks, in Sections 4.5.1, 4.5.2, and 4.5.3, respectively. We report the results of our experiments in Sections 4.6.1, 4.6.2 and 4.6.3. Finally, we present the analyses in Section 4.7.

4.2 Related Work

Researchers have often aimed to reduce the memory consumption of word embeddings in the real world because a relatively large memory is required. Suzuki and Nagata (Suzuki and Nagata, 2016) proposed a parameter reduction method for word embeddings that used machine learning techniques. Our method can also be interpreted as a type of parameter reduction method based on subword features. However, their method only considers model shrinkage and does not utilize any subword information or consider the OOV issue.

The OOV word issue is a widely discussed topic in word embedding research that several researchers have recently attempted to solve. For example, methods that leverage subword information, such as character N-grams (including character unigrams) (Bojanowski et al., 2017; Pinter et al., 2017; Zhao et al., 2018), and morphological features (Luong et al., 2013), have recently been discussed as means of constructing word embeddings that consider the applicability of OOV words. Moreover, `SemLand` (Pilehvar and Collier, 2017) and `ALIGN` (Prokhorov et al., 2019) have been proposed, which induce OOV word embeddings by leveraging external resources. Similar methods that estimate OOV word embeddings by using an additional LSTM (Bahdanau et al., 2017) and leveraging a small additional dataset (Herbelot and Baroni, 2017) have also been recently proposed.

Among these studies, the study most closely related to ours is that of Zhao et al. (Zhao et al., 2018). Their basic idea is to reconstruct each pre-trained word embedding by using bag-of-character N-grams. We refer to their method as "`BoS`." The motivation for reconstructing pre-trained word embeddings and utilizing character N-grams in our approach is substantially the same; however, an essential difference from `BoS` is that we additionally consider jointly reducing the total number of embedding vectors.

Another study with the same motivation and goal is that of Pinter et al. (Pinter et al., 2017). Their method, referred to as `MIMICK`, utilizes only character information instead of character N-grams by mixing it with a more sophisticated neural network, namely, LSTM (Hochreiter and Schmidhuber, 1997). `MIMICK` can produce a more compact model than the original word embeddings can. The important difference between their method and ours is that our method only consists of subword embeddings, whereas their method consists of character embeddings and several transformation matrices for calculating LSTMs. We compare their method with ours in our experiments and empirically show the effectiveness of our approach.

Moreover, Bojanowski et al. (Bojanowski et al., 2017) proposed a method called `fastText`, which also incorporates character N-gram embeddings in addition to word embeddings. However, they did not explicitly prove the effectiveness of OOV word embeddings. Thus, how well the combination of character N-grams can reconstruct appropriate embeddings for OOV words remains unclear. In addition, their method trains word embeddings from a corpus, which is not a reconstruction setting discussed in this paper. Therefore, we consider their method orthogonal to ours.

For neural language models (LMs), Labeau and Allauzen (Labeau and Allauzen, 2017) investigated a setting: using full word embeddings for frequent words and character-based embeddings for less frequent words. This setting can be viewed as simultaneously achieving a smaller number of memory requirement and a higher applicability of OOV words. However, they did not explore the relation between memory requirement and the applicability of OOV words, because the main purpose of their work was to investigate the effect on a language model's performance when using the subword embeddings in output layers of the LMs.

In summary, in the context of obtaining word embeddings, no study has attempted to simultaneously achieve a smaller number of embedding vectors and a higher applicability of OOV words; thus, this paper is the first attempt to investigate how these two aims can be simultaneously achieved.

Additionally, deep contextualized pre-trained LMs, such as ELMo (Peters et al., 2018a), BERT (Devlin et al., 2019), and several variants of them have recently been proposed as alternatives to the pre-trained word embeddings to further improve task performances. However, some pre-trained LMs, such as ELMo, continue to take advantage of pre-trained word embeddings to achieve their state-of-the-art performance. Moreover, up-to-date contextualized embeddings have been demonstrated to be insufficient for handling rare words, and learning embeddings specifically for rare words could still be beneficial (Schick and Schütze, 2020; Schick and Schütze, 2020). This finding implies that pre-trained word embeddings can still be combined with strong pre-trained LMs; thus, the importance of word embeddings in the literature remains unchanged, although stronger pre-trained models have been established.

4.3 Reconstruction of Word Embeddings Using Subwords

In this section, we formally define the task we tackled in this paper.

4.3.1 Notation Rules

We use the following notation rules unless otherwise specified.

1. A lower-case bold letter (e.g., $\boldsymbol{v}$ or $\boldsymbol{e}$) represents a column vector.
2. An upper-case bold letter (e.g., $\boldsymbol{V}$ or $\boldsymbol{E}$) represents a matrix.
3. An upper-case letter in calligraphy form (e.g., $\mathcal{W}$ or $\mathcal{S}$) denotes a set.
4. A Greek letter (e.g., Φ or η) indicates a function.
5. A lower-case letter (e.g., z or i) denotes a scalar variable, index, or symbol.
6. An upper-case letter (e.g., C or H) indicates a scalar hyper-parameter.
7. The notation $\boldsymbol{V}[i]$ is introduced to represent the i-th column vector in the matrix $\boldsymbol{V}$ to simplify the representation.
8. $||\boldsymbol{v}||_p$ represents the L_p-norm of the given vector $\boldsymbol{v}$.
9. The absolute value of a set, such as $|\mathcal{W}|$ or $|\mathcal{S}|$, indicates the number of instances in the corresponding set.

4.3.2 Preliminaries

Words and their embedding

We define a mapping function $\zeta(\cdot)$ from the word space to the corresponding index space as follows:

$$\zeta(\cdot) : \mathcal{W} \to \mathcal{I}_{|\mathcal{W}|}, \tag{4.1}$$

where $\mathcal{W}$ is a set of words (vocabulary), and $\mathcal{I}_{|\mathcal{W}|}$ is a set of integers that represents the set of indices of word embeddings. In this paper, we define that $\zeta(\cdot)$ is a bijective function; thus, each word has its own unique index between 1 and $|\mathcal{W}|$. This also implies that the relation $|\mathcal{W}| = |\mathcal{I}_{|\mathcal{W}|}|$ always holds. Let $\boldsymbol{e}_w$ be a D-dimensional embedding vector for the word $w \in \mathcal{W}$, and $\boldsymbol{E} \in \mathbb{R}^{D \times |\mathcal{I}_{|\mathcal{W}|}|}$ denotes an embedding matrix for all words in $\mathcal{W}$. We then assume that the following relation always holds between $\boldsymbol{e}_w$ and $\boldsymbol{E}$:

$$\boldsymbol{e}_w = \boldsymbol{E}[i], \quad \text{where} \quad i = \zeta(w). \tag{4.2}$$

Therefore, the i-th column vector in the matrix $\boldsymbol{E}$ represents the word embedding of the corresponding word w that satisfies $i = \zeta(w)$.

Subwords and their embedding

Let S be a vocabulary for all pre-defined subwords obtained from the words in W. We assume that the subwords are the character N-grams that appeared in $w \in W$. Similar to Eq. 4.1, we also define a mapping function $\eta(\cdot)$ from the subword space to the corresponding index space as follows:

$$\eta(\cdot) : S \to I_{|S|}, \tag{4.3}$$

where $I_{|S|}$ is a set of integers that represents the set of indices of subword embeddings. As well as $\zeta(\cdot)$ in Eq. 4.1, $\eta(\cdot)$ is generally defined as a bijective function.[4] In this case, each subword has its own unique index between 1 and $|S|$, and the relation $|S| = |I_{|S|}|$ always holds. We also assume the following relation between $\boldsymbol{v}_s$ and $\boldsymbol{V}$:

$$\boldsymbol{v}_s = \boldsymbol{V}[j], \quad \text{where} \quad j = \eta(s), \tag{4.4}$$

where $\boldsymbol{v}_s$ is a D-dimensional vector for the subword $s \in S$, and $\boldsymbol{V} \in \mathbb{R}^{D \times |I_{|S|}|}$ is an embedding matrix for all subwords in S. Therefore, the j-th column vector in the matrix $\boldsymbol{V}$ represents the subword embedding of the corresponding subword s that satisfies $j = \eta(s)$.

Word to subword mapping

Let L represent a set of the lists that consists of $s \in S$. We then introduce a mapping function $\phi(\cdot)$ as follows:

$$\phi(\cdot) : W \to L. \tag{4.5}$$

For example, if we specifically define S as all the character bi-grams appearing in $w \in W$, then $\phi(w)$, where $w =$ "newer", returns a list of seven distinct subword indices of "⟨w⟩n", "ne", "ew", "we", "er", and "r⟨/w⟩", where "⟨w⟩" and "⟨/w⟩" are special characters that represent the beginning and end of a word, respectively.

[4] We redefine η as a surjective function in Section 4.4.2.

4.3.3 Task Definition

Conceptually, we aim to reconstruct all the embeddings in $\boldsymbol{E}$ using $\boldsymbol{V}$ and a pre-defined subword mixing function $\tau(\cdot)$. Formally, our reconstruction problem is represented as a minimization problem of the following form:

$$\widehat{\boldsymbol{V}} = \operatorname*{arg\,min}_{\boldsymbol{V}} \left\{ \Psi(\boldsymbol{E}, \boldsymbol{V}, \tau) \right\}, \tag{4.6}$$

where $\Psi(\cdot)$ is the loss function used to calculate the total reconstruction loss between $\boldsymbol{E}$ and $\boldsymbol{V}$. In summary, our goal is to find $\widehat{\boldsymbol{V}}$ for which the loss function $\Psi(\cdot)$ is minimized from a machine learning perspective.

4.3.4 Baseline Method

Subword mixing function $\tau(\cdot)$

The role of the function $\tau(\cdot)$ is to calculate an alternative embedding of the word embedding $\boldsymbol{e}_w$ using a list of subwords contained in the given word w. One of the most popular definitions of $\tau(\cdot)$ is to simply sum all the obtained subwords as follows:

$$\tau_{\mathtt{sum}}(\boldsymbol{V}, w) = \sum_{s \in \phi(w)} \boldsymbol{v}_s. \tag{4.7}$$

A subword mixing function of this form has been used in the previous methods, such as `fastText` and `BoS`.

Loss function $\Psi(\cdot)$

First, to improve readability, we introduce $\hat{\boldsymbol{v}}_w$ as an abbreviated notation of $\tau(\boldsymbol{V}, w)$, namely,

$$\hat{\boldsymbol{v}}_w = \tau(\boldsymbol{V}, w). \tag{4.8}$$

Several choices are possible for the definition of the loss function $\Psi(\cdot)$. We consider using a squared loss function, which can be written as the summation of the squared losses over an individual embedding vector $\boldsymbol{e}_w$:

$$\Psi(\boldsymbol{E}, \boldsymbol{V}, \tau) = \sum_{w \in W} C_w \left\| \boldsymbol{e}_w - \hat{\boldsymbol{v}}_w \right\|_2^2, \tag{4.9}$$

Table 4.1 Statistics for each setting: The column "# of vecs" represents the number of embedding vectors. "mem." represents the memory requirement to store the vectors and the indexes between words and their vectors. M denotes 1 million. We consider that a real value requires 4 bytes of storage in the calculation of the memory requirement.

ID	Setting		# of vecs	mem. (GB)
`(a)`	original `fastText.600B`		2.0 M	2.3 GB
`(b)`	char N-gram	$N = 1,2,3$	0.2 M	0.2 GB
`(c)`		$N = 3,4,5,6$	6.2 M	7.0 GB
`(d)`		$N = 1$ to 6	6.3 M	7.1 GB
`(e)`		$N = 1$ to ∞	24.9 M	28.1 GB
`(f)`	original `GloVe.840B`		2.2 M	2.5 GB
`(g)`	char N-gram	$N = 1,2,3$	0.2 M	0.2 GB
`(h)`		$N = 3,4,5,6$	7.1 M	8.0 GB
`(i)`		$N = 1$ to 6	7.1 M	8.0 GB
`(j)`		$N = 1$ to ∞	30.6 M	34.6 GB

where C_w is a logarithm of the frequency of word w in a corpus. Intuitively, $\Psi(\cdot)$ calculates the weighted sum of the L_2-norm distances between the reference vector $\boldsymbol{e}_w$ and a vector calculated by using the subword mixing function $\tau(\cdot)$.

4.3.5 Consideration of Memory Requirement

An important aspect to consider in our subword-based approach is the number of embedding vectors. Table 4.1 presents the total number of embedding vectors and the total memory requirement in several settings. The original `fastText.600B` word embeddings consist of 2 million words. Each word embedding is a $D = 300$ dimensional vector and thus requires 2.3 GB of storage (row `(a)`). Breaking these down into character N-grams allows for a greater flexibility leading to the successful handling of OOV words, however, the memory requirement greatly varies depending on N. For example, if we were to use every possible character N-gram obtained from every word (row `(e)`), the memory requirement increases to approximately 28 GB, which is too large for practical use. Therefore, finding a balanced approach to technically reduce the memory requirement is crucial.

A fairly straightforward approach is to only use a range of smaller N-grams, such as $N = 1$ to 3 (row `(b)`) or $N = 3$ to 6 (row `(c)`). However, a smaller subword setting (e.g., `(b)`) might lag behind the original word embeddings in terms of performance. The optimal setting would therefore be when the number of vectors (i.e., the memory requirement) is smaller than the original embeddings while maintaining a comparable performance to theirs.

4.4 Methods to Improve Performance

To achieve the goal of a lower memory requirement while maintaining original performance, we introduce several modifications to the baseline word embedding reconstruction approach explained in Section 4.3. Basically, we enhance the mapping function $\eta(\cdot)$ and the subword mixing function $\tau(\cdot)$.

4.4.1 Frequent Subwords

Instead of all possible subwords, we use the top-F frequent subwords that can be taken from the words in vocabulary W. Let S_F represent the set of the top-F frequent subwords ($F = |S_F|$), where $S_F \subseteq S$. We define a new mapping function $\eta_F(\cdot)$ as follows:

$$\eta_F(\cdot) : S_F \rightarrow I_F. \tag{4.10}$$

Hereafter, we assume that S is substituted by S_F when we calculate $\phi(\cdot)$ for $\eta_F(\cdot)$. This means that $\phi(\cdot)$, which is a word to subword mapping, returns only the top-F frequent subwords.

4.4.2 Memory Sharing

In the baseline method, we assume that the mapping function $\eta(\cdot)$ is a bijective function as described in Section 4.3.2. Again, this means that each subword has its own unique subword index. We modify $\eta(\cdot)$ to a surjective function by introducing the following new mapping function $\eta_H(\cdot)$ as a replacement for $\eta(\cdot)$ in Eq. 4.3:

$$\eta_H(\cdot) : S \rightarrow I_H. \tag{4.11}$$

Here, we assume $H < |S|$. This mapping function $\eta_H(\cdot)$ implies that each subword is mapped to an index, but the index is not unique and may be shared with other subwords. Therefore, the number of subword embeddings can also be reduced to H by sharing the embeddings; thus, the subword embedding matrix $\boldsymbol{V}$ can also be reduced from $\boldsymbol{V} \in \mathbb{R}^{D\times|S|}$ to $\boldsymbol{V} \in \mathbb{R}^{D\times H}$. If we assume the relation $H \ll |S|$, we can greatly reduce the total embedding size.

Despite the several possible choices for the definition of the mapping function $\eta_H(\cdot)$, we use a hash function which takes a string and returns a random integer from uniform distribution. We simply consider the remainder of a hash value divided by H as an index of a subword. Thus, subword embeddings are randomly shared over the subwords. Our models optimize their embedding parameters so that the collisions are well tolerated. An advantage of using simple hash functions is that an external mapping structure is not required for every

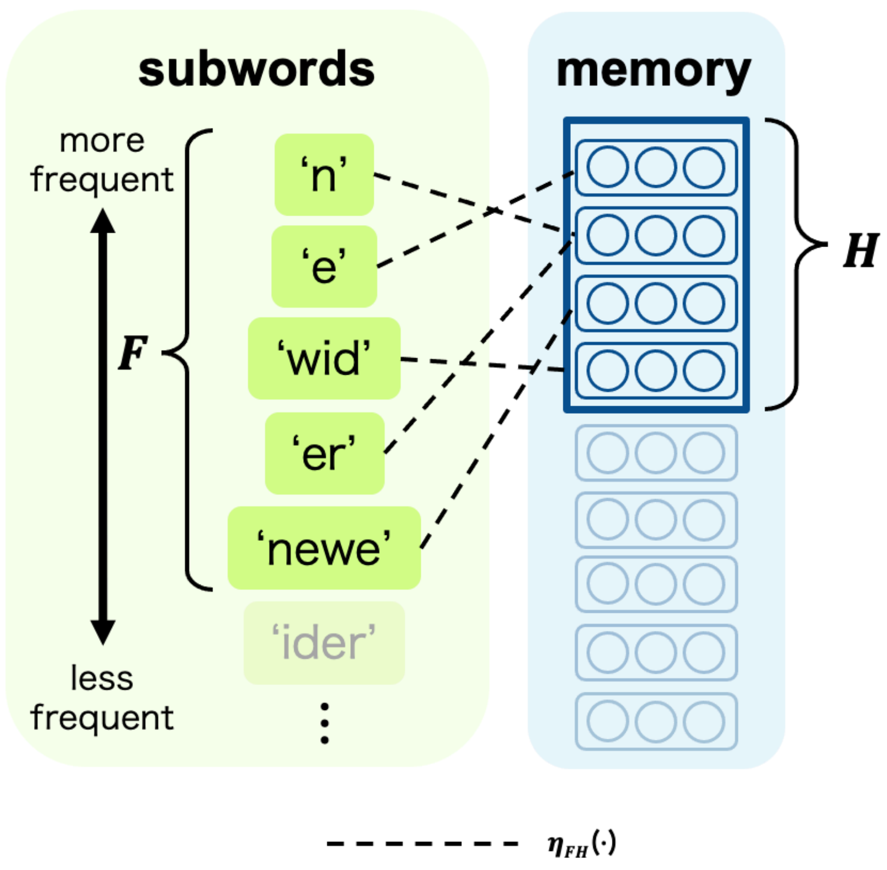


Figure 4.1 Intuitive idea for the combination of frequent subwords and memory-shared embeddings.

(subword, subword index) pair, which also matches our goal of reducing the memory requirement in actual use cases. Specifically, following a previous study, FastText (Bojanowski et al., 2017), we use a Fowler–Noll–Vo (FNV)[5] hash function.

4.4.3 Combination of Frequent Subwords and Memory Sharing

$\eta_F(\cdot)$ and $\eta_H(\cdot)$ can also be combined step by step. We first reduce the subword vocabulary S to the top-F frequent subwords S_F as described in Section 4.4.1. We then apply our memory sharing method to only S_F, in contrast to applying it to S in Section 4.4.2. Here, we define a new mapping function $\eta_{FH}(\cdot)$ as follows:

$$\eta_{FH}(\cdot) : S_F \rightarrow I_H. \tag{4.12}$$

[5]http://www.isthe.com/chongo/tech/comp/fnv/index.html

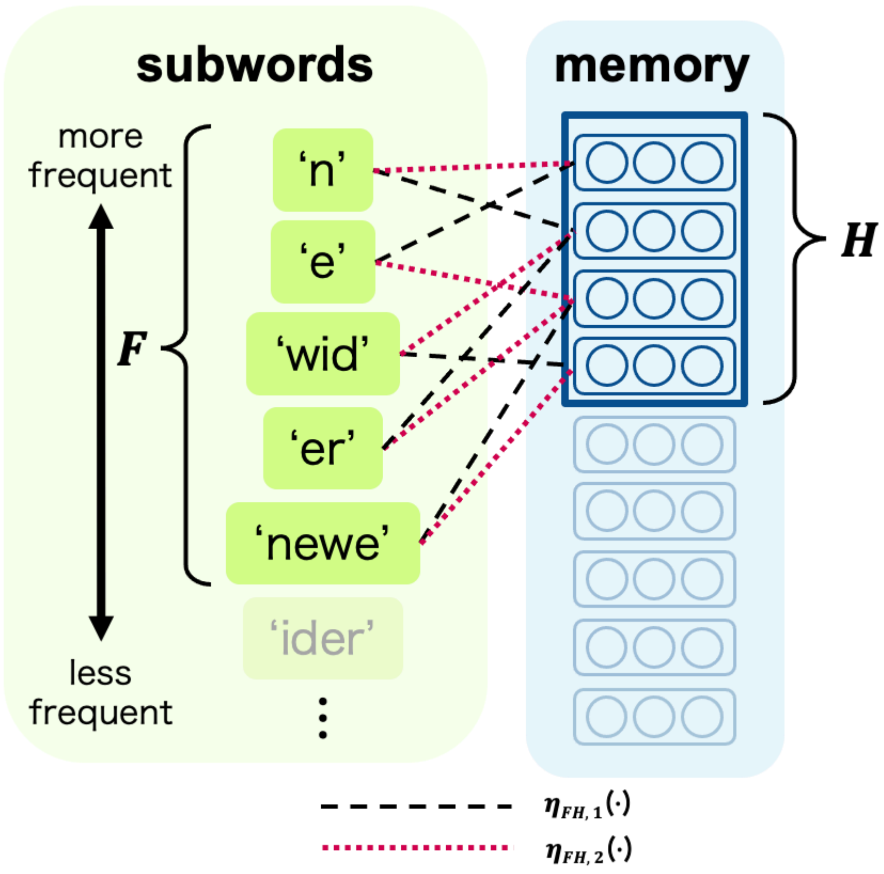


Figure 4.2 Comparison of using single hash function (Fig. 4.1) with the multiple hash function case. This figure shows the case when P (the number of hash functions) is set to 2.

Similar to $\eta_F(\cdot)$, we use S_F to calculate $\phi(\cdot)$ for $\eta_{FH}(\cdot)$ instead of S. Moreover, we assume $F \geq H$. Note that Eq. 4.12 becomes identical to Eq. 4.11, Eq. 4.10, and Eq. 4.3 if we set $F = |S|$, $H = |S_F|$, and $H = F = |S|$, respectively. An intuitive idea for the combination of frequent subwords and memory-shared embeddings is illustrated in Fig. 4.1.

4.4.4 Attention Operation

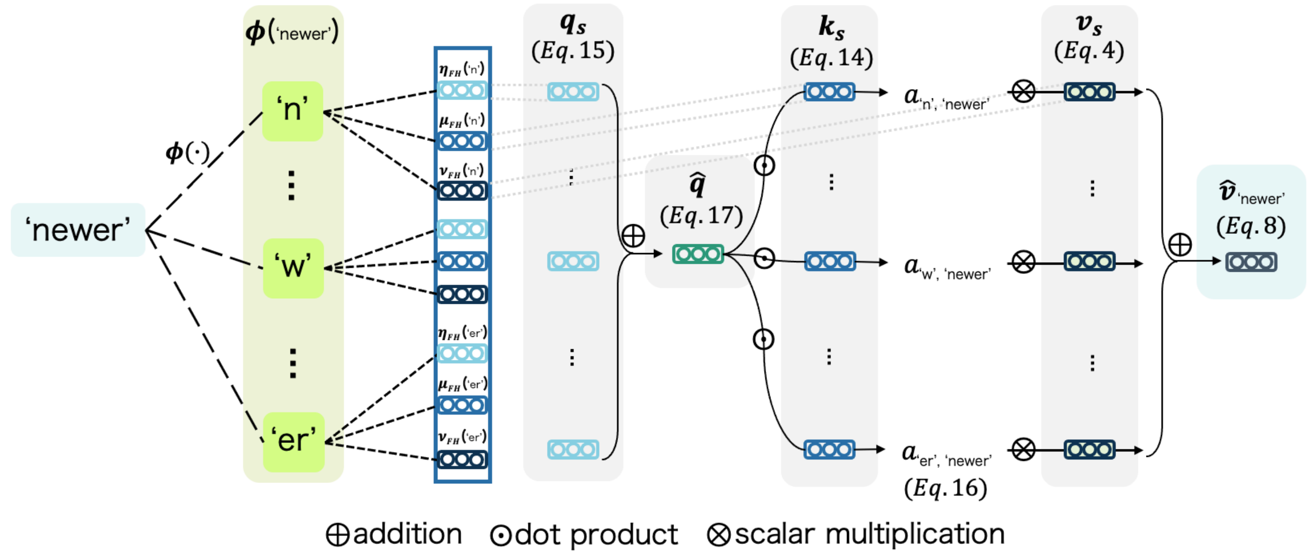


Figure 4.3 Illustration of how our KVQ self-attention operation calculates each word embedding. This example shows a process of obtaining a word embedding for a word "newer" by using the KVQ self-attention operation. First, we input the word "newer" into $\phi(\cdot)$, a word to subword mapping function, to obtain the subwords of "newer" (green box). Next, we use mapping functions, e.g., $\eta(\cdot)$, to obtain the indexes of those subwords and pick the corresponding embeddings (blue table). Next, we calculate each weight α using the query and key embeddings and then calculate the weighted sum of the value embeddings (gray box) to obtain the final embedding of "newer."

Previous methods such as `fastText` and `BoS` treat $\tau(\boldsymbol{V}, w)$ as a summation of all subword embeddings described by Eq. 4.7. However, the summation is less expressive and might not have capability in a memory-sharing setting because subwords share their embeddings randomly. One possible improvement is to handle the importance of each subword based on a given word during the calculation of $\Phi(\boldsymbol{V}, w)$. A simple approach to realize this improvement is to incorporate a "context-dependent" weighting factor for each subword in a given word.

Thus, we consider the following subword mixing function:

$$\tau_{\mathtt{kvq}}(\boldsymbol{V}, w) = \sum_{s \in \phi(w)} a_{s,w} \boldsymbol{v}_s, \tag{4.13}$$

where $a_{s,w}$ represents a context-dependent weighting factor of the subword s, where the "context" means all the subwords obtained from $\phi(w)$.

To calculate $a_{s,w}$, we first introduce $\boldsymbol{k}_s$ and $\boldsymbol{q}_s$, which are defined similarly to $\boldsymbol{v}_s$ in Eq. 4.4, namely,

$$\boldsymbol{k}_s = \boldsymbol{V}[m], \quad \text{where} \quad m = \mu_{FH}(s), \tag{4.14}$$

and

$$\boldsymbol{q}_s = \boldsymbol{V}[n], \quad \text{where} \quad n = \nu_{FH}(s). \tag{4.15}$$

Similar to $\eta_{FH}(\cdot)$ in Eq. 4.12, $\mu_{FH}(\cdot)$ and $\nu_{FH}(\cdot)$ are two distinct mapping functions that map a given subword s into a subword index. Next, we introduce a key-value-query (KVQ) self-attention operation inspired by `Transformer` (Vaswani et al., 2017):

$$a_{s,w} = \frac{\exp(Z\hat{\boldsymbol{q}} \cdot \boldsymbol{k}_s)}{\sum_{s' \in \phi(w)} \exp(Z\hat{\boldsymbol{q}} \cdot \boldsymbol{k}_{s'})}, \tag{4.16}$$

$$\hat{\boldsymbol{q}} = \sum_{s \in \phi(w)} \boldsymbol{q}_s, \tag{4.17}$$

where Z is a scaling hyper-parameter. Overall, Fig. 4.3 illustrates how our KVQ self-attention operation calculates each word embedding.

4.4.5 Incorporating Multiple Hash Functions

We can further enhance the memory-sharing method by introducing multiple hash functions. Let P be the number of distinct hash functions. We can, for example, straightforwardly

Table 4.2 Evaluation datasets used in our experiments.

abbre.	data size	number of OOV data `fastText.600B`	number of OOV data `GloVe.840B`
Word similarity estimation (**WordSim**)			
MEN (Bruni et al., 2014)	3,000	0	0
M&C (Miller and Charles, 1991)	30	0	0
MTurk (Radinsky et al., 2011)	287	0	0
RW (Luong et al., 2013)	2,034	37	36
R&G (Rubenstein and Goodenough, 1965)	65	0	0
SCWS (Huang et al., 2012)	2,003	2	2
SLex (Hill et al., 2014)	998	0	0
WSR (Agirre et al., 2009)	252	0	0
WSS (Agirre et al., 2009)	203	0	0
Word analogy estimation (**Analogy**)			
GL (Mikolov et al., 2013a)	19,544	0	0
MSYN (Mikolov et al., 2013d)	8,000	1000	1000

prepare P distinct hash functions, without changing any hash algorithm, by introducing P different random seeds when we calculate the hash functions.

An advantage of introducing multiple hash functions is that we can significantly reduce the probability of assigning an identical index set to multiple subwords. For instance, if we suppose that each hash function returns a fully random index given an input s, the probability of assigning a certain index set can be estimated as $(1/H)^P$. Therefore, a larger P may alleviate the negative effect of hash collision.

To incorporate multiple hash functions into our method, we reformulate some equations by adding an index p, where $1 \leq p \leq P$. to several components. First, we substitute $a_{s,w}$, $\boldsymbol{v}_s$, $\boldsymbol{k}_s$ $\boldsymbol{q}_s$ with $a_{s,w,p}$, $\boldsymbol{v}_{s,p}$, $\boldsymbol{k}_{s,p}$, and $\boldsymbol{q}_{s,p}$, respectively. Second, we add another summation of p inside the summation of $s \in \phi(w)$ in Eqs. 4.13, 4.16, and 4.17. For example, Eq. 4.13 is reformulated to the following equation for utilizing a multiple hash function case:

$$\tau_{\mathtt{kvq}}(\boldsymbol{V}, w) = \sum_{s \in \phi(w)} \sum_{1 \leq p \leq P} a_{s,w,p} \boldsymbol{v}_{s,p}, \tag{4.18}$$

Third, we introduce $\eta_{FH,p}(\cdot)$, $\mu_{FH,p}(\cdot)$ and $\nu_{FH,p}(\cdot)$ as the proxy of $\eta_{FH}(\cdot)$ in Eq. 4.12, $\mu_{FH}(\cdot)$ in Eq. 4.14, and $\nu_{FH}(\cdot)$ in Eq. 4.15, respectively, to explicitly distinct P hash functions. For example, Eq. 4.14 is rewritten as follows:

$$\boldsymbol{k}_{s,p} = \boldsymbol{V}[m], \quad \text{where} \quad m = \mu_{FH,p}(s), \tag{4.19}$$

Table 4.3 Model and training settings of the model shrinkage experiments.

Model setting	Word embedding (dimension)	$D = 300$
	Scaling parameter in Eq. 4.16	$Z = \sqrt{D}$
Training setting	Optimizer	Adam
	Options of Adam	$\alpha = 0.0001$ $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\epsilon = 1 \times 10^{-8}$
	Mini-batch size	200
	# of epochs	300

Fig. 4.2 illustrates the case of using multiple hash functions for $\eta_{FH,p}(\cdot)$. The difference between single and multiple hash function cases is found by comparing Figs. 4.1 and 4.2.

It is worth noting here that there is a limitation for increasing P; as the number of hash functions increases, so does the possibility of collisions; i.e., the total amount of subwords being mapped to the same memory segment. Thus, we need to explore the appropriate P that balances the positive effect (increasing distinguishability of each subword) and the negative effect (increasing the collision probability) of increasing the number of hash functions.

4.5 Experimental Settings

We analyze the benefits of our method by using three sets of experiments: model shrinkage, OOV word embeddings, and downstream tasks. The model shrinkage experiments demonstrate how our models reduce the memory requirement while maintaining the original performance. The OOV word embedding experiments confirm that our learned embeddings can be applied to OOV words. The experiments on downstream tasks confirm that our reduced embeddings can still be useful for real-world applications through extrinsic evaluation tasks such as named entity recognition (NER) and textual entailment (TE). The following sections contain detailed descriptions of the experimental settings for each set of experiments.

4.5.1 Model Shrinkage

We compare five distinct settings of subword-based reconstruction of word embeddings:

1. **SUM-F**: Select $\eta_F(\cdot)$ in Eq. 4.10 (Section 4.4.1) for the subword mapping function and $\tau_{\mathtt{sum}}(\cdot)$ in Eq. 4.7 for the subword mixing function.

2. **SUM-H**: As in the first setting but substitute $\eta_F(\cdot)$ with $\eta_H(\cdot)$ in Eq. 4.11 (Section 4.4.2).

3. **KVQ-H**: As in the second setting but substitute $\tau_{\mathtt{sum}}(\cdot)$ in Eq. 4.7 with $\tau_{\mathtt{kvq}}(\boldsymbol{V}, w)$ in Eq. 4.13 (Section 4.4.4).

4. **SUM-FH**: As in the second setting but substitute $\eta_H(\cdot)$ with $\eta_{FH}(\cdot)$ in Eq. 4.12 (Section 4.4.3).

5. **KVQ-FH**: As in the third setting but substitute $\eta_H(\cdot)$ with $\eta_{FH}(\cdot)$ in Eq. 4.12 (Section 4.4.3).

Each setting is tested on well-studied linguistic benchmark tasks, namely, nine for word similarity (**WordSim**) tasks and two for word analogy (**Analogy**) tasks. The evaluation datasets used in our experiments are summarized in Table 4.2. A unit of data is a pair of words whose similarity needs to be measured in the WordSim test; by contrast, in the Analogy test, a sample consists of a list of four words, e.g., ['king' , 'queen' , 'man' , 'woman']. We call such samples that contain at least one OOV word as "OOV data" .

Following the standard evaluation criteria used in previous studies, we discard OOV data in the evaluation datasets.

Although our models can calculate OOV word embeddings, we enforce this restriction for a fair comparison to the original (word-based) word embeddings.

For the reconstruction target, namely $\boldsymbol{E}$ in Eq. 4.2, we select `fastText.600B` or `GloVe.840B`. Note that `fastText.600B` achieved state-of-the-art performance on the WordSim and Analogy datasets (Bojanowski et al., 2017). The hyper-parameters $D = 300$ and $|\mathcal{W}|$ are automatically obtained from the properties of `fastText.600B` and `GloVe.840B`. For C_w in Eq. 4.9, we utilize the occurrence information calculated from a large external corpus. We apply a function that returns a set of all the character N-grams, where $N = 3$ to 30, of a given word as the subword mapping function $\phi(\cdot)$ defined in Eq. 4.5. The remaining hyper-parameters are presented in Table 4.3.

Additionally, as preliminary experiments of the hyper-parameter search, we explore the optimal P for each memory-shared method, namely, `SUM-H`, `KVQ-H`, `SUM-FH` and `KVQ-FH`, individually and independently from the range of $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$. Here we select a single P per method. This means that we use the selected P regardless of the shared memory size H, tasks such as WordSim, and the vector types such as `fastText.600B`.

Concretely, we calculate overall performances of models among various Hs and tasks (WordSim and Analogy) as follows:

$$\mathtt{U}^p_{\mathtt{overall}} = \frac{\mathtt{U}^p_{\mathtt{WordSim}} + \mathtt{U}^p_{\mathtt{Analogy}}}{2}, \tag{4.20}$$

$$\mathtt{U}^p_{\mathtt{WordSim}} = \frac{\sum_{h \in H} \mathtt{U}^{p,h}_{\mathtt{WordSim}}}{|H|}, \mathtt{U}^p_{\mathtt{Analogy}} = \frac{\sum_{h \in H} \mathtt{U}^{p,h}_{\mathtt{Analogy}}}{|H|}, \tag{4.21}$$

where $H = \{0.2\text{M}, 0.3\text{M}, 0.4\text{M}, 0.5\text{M}\}$, a set of the size H for memory sharing. $\mathtt{U}^{p,h}$ is the evaluation score of each method that is normalized in the range of [0,1]. To obtain a vector-independent P, we first compute $\mathtt{U}^p_{\mathtt{overall}}$ for `fastText.600B` and $\mathtt{U}^p_{\mathtt{overall}}$ for `GloVe.840B`, respectively, and then average them. Here, we assume that such preliminary experiments are the development set for the P selection. We select the P with which a model has the best overall performance. We use the obtained Ps in the remaining OOV (Section 4.5.2, 4.6.2) and downstream tasks (Section 4.5.3, 4.6.3).

4.5.2 OOV Word Embeddings

As demonstrated in Table 4.2, the numbers of OOV problems for `fastText.600B` and `GloVe.840B` is very small because their vocabulary sizes exceed 2 million words; moreover, the words contained in the evaluation datasets tend to be "non-rare words" in general. Therefore, it is difficult to precisely evaluate the effectiveness of the estimation of OOV word embeddings. To overcome this issue, we conduct a synthetic OOV experiment in addition to non-synthetic ones from previous studies.

Synthetic OOV experiment

In a synthetic OOV experiment, we artificially form OOV words in the reconstruction problem, namely, we discard the words in the evaluation datasets from the vocabulary W for training. We suppose that $(w^{\mathrm{a}}_i, w^{\mathrm{b}}_i)$ represents the i-th evaluation instance of a WordSim dataset that consists of a word pair w^{a}_i and w^{b}_i. Next, we let D be the WordSim dataset, that is, $(w^{\mathrm{a}}_i, w^{\mathrm{b}}_i) \in D$. We build a word set W_b by extracting one word from each evaluation instance, namely, $W_b = \{w^{\mathrm{b}}_i | (w^{\mathrm{a}}_i, w^{\mathrm{b}}_i) \in D, 1 \leq i \leq |D|\}$. Next, we eliminate the word embeddings corresponding to the words in W_b from the reconstruction target embeddings such as `fastText.600B`. As a result, we obtain a subset of embeddings and use them as the target embeddings of our reconstruction strategy instead of the original ones. We can

Table 4.4 Statistics for our methods.

method	F	H	$\|W\|$	$\|S\|$	size (GB)
`fastText.600B`			2M	–	2.26GB
`SUM-F`	0.5M	-	2M	0.5M	0.59GB
`SUM-H`	-	0.5M	2M	24.9M	0.59GB
`KVQ-H`	-	0.5M	2M	24.9M	0.59GB
`SUM-FH`	1.0M	0.5M	2M	1.0M	0.59GB
`KVQ-FH`	1.0M	0.5M	2M	1.0M	0.59GB
`SUM-F`	0.2M	-	2M	0.2M	0.23GB
`SUM-H`	-	0.2M	2M	24.9M	0.23GB
`KVQ-H`	-	0.2M	2M	24.9M	0.23GB
`SUM-FH`	1.0M	0.2M	2M	1.0M	0.23GB
`KVQ-FH`	1.0M	0.2M	2M	1.0M	0.23GB

method	F	H	$\|W\|$	$\|S\|$	size (GB)
`GloVe.840B`			2.2M	–	2.48GB
`SUM-F`	0.5M	-	2.2M	0.5M	0.59GB
`SUM-H`	-	0.5M	2.2M	30.6M	0.59GB
`KVQ-H`	-	0.5M	2.2M	30.6M	0.59GB
`SUM-FH`	1.0M	0.5M	2.2M	1.0M	0.59GB
`KVQ-FH`	1.0M	0.5M	2.2M	1.0M	0.59GB
`SUM-F`	0.2M	-	2.2M	0.2M	0.23GB
`SUM-H`	-	0.2M	2.2M	30.6M	0.23GB
`KVQ-H`	-	0.2M	2.2M	30.6M	0.23GB
`SUM-FH`	1.0M	0.2M	2.2M	1.0M	0.23GB
`KVQ-FH`	1.0M	0.2M	2.2M	1.0M	0.23GB

consider the words in W_b to be synthetic OOV words because our models do not see the embeddings corresponding to the words in W_b in the training phase. Also, we process the datasets of word analogy tasks in the same manner. Thus, all the problems in the evaluation datasets become OOV problems. In other words, the amount of OOV data in Table 4.2 in this setting always equals the amount of evaluation data, such as 2034 for RW. Here we used the same experimental settings as in Section 4.5.1 unless otherwise specified.

Comparison with previous studies

As discussed in Section 4.2, several closely related methods have also been used in attempts to solve the OOV word issue, such as `MIMICK` and `BoS`. To directly compare our approach with these methods, we strictly follow the experimental settings described in the previous study (Zhao et al., 2018) and compare the performance under fair conditions. In the evaluation setting, the OOV word performance is evaluated over RW (Table 4.2), where all the words appearing in RW were included as the evaluation data, in contrast to discarding the OOV data in Section 4.5.1.

The target word embeddings for the reconstruction were the embeddings trained on Google News with 100 billion tokens[6] that were pre-cleaned by the previous study (Zhao et al., 2018). The resultant embeddings consist of 0.16M lower-cased word embeddings. We compare our approach with the following related methods:

1. `Random`: the performance when random vectors are used for OOV words.
2. `MIMICK`[7] (Pinter et al., 2017).
3. `BoS`[8] (Zhao et al., 2018).

For our reconstruction model, the shared memory size H was set to 0.04M because the vocabulary of this setting is relatively small compared with that in our experiments, namely, 2M vs 0.16M. For the other settings, for example, subword mapping function $\phi(\cdot)$, we use the same settings as in Section 4.5.1.

[6]https://code.google.com/archive/p/word2vec/
[7]https://github.com/yuvalpinter/Mimick
[8]https://github.com/jmzhao

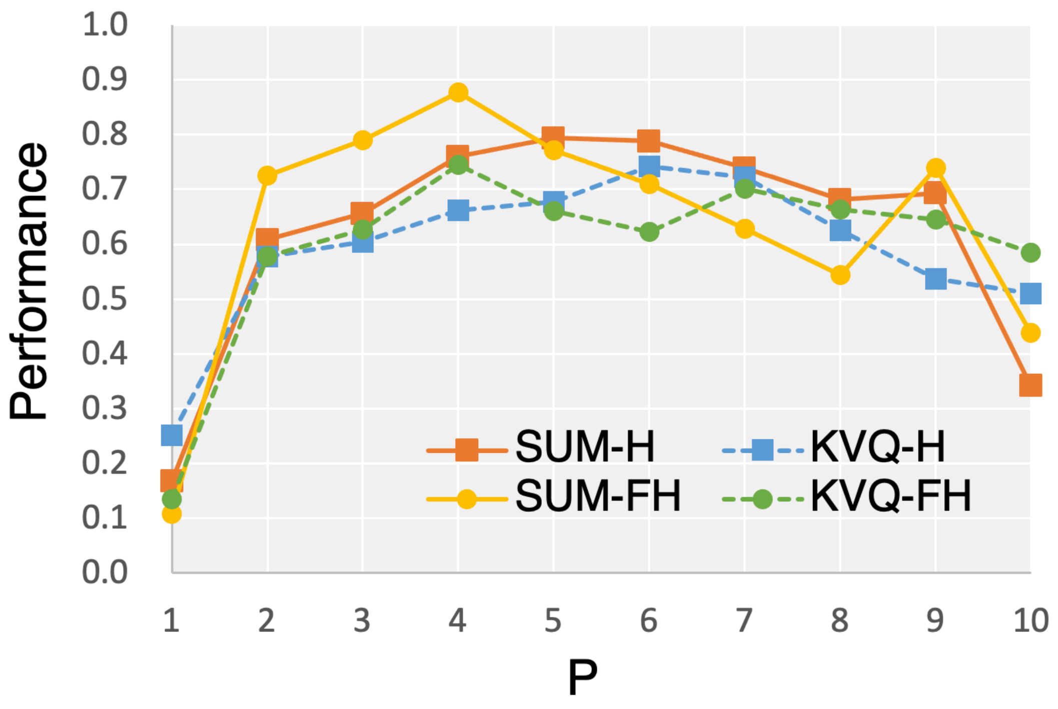
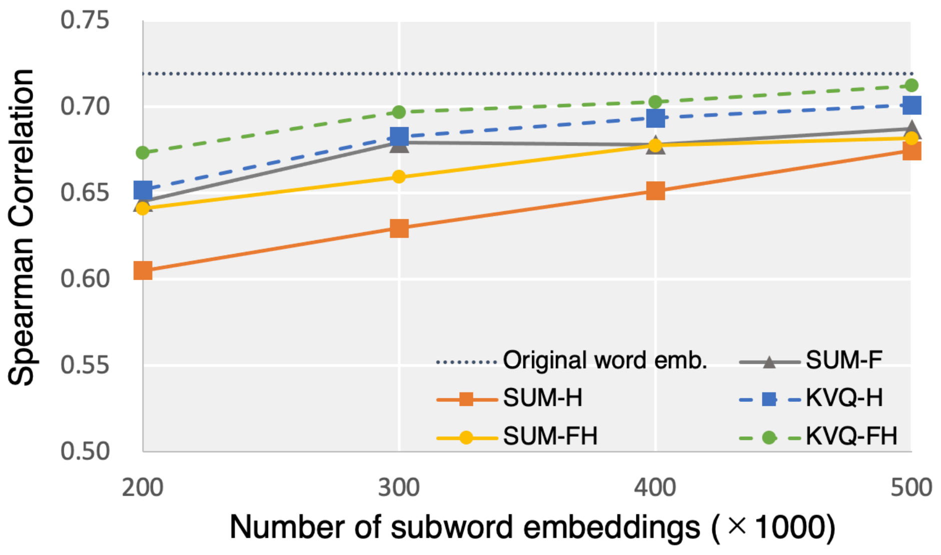
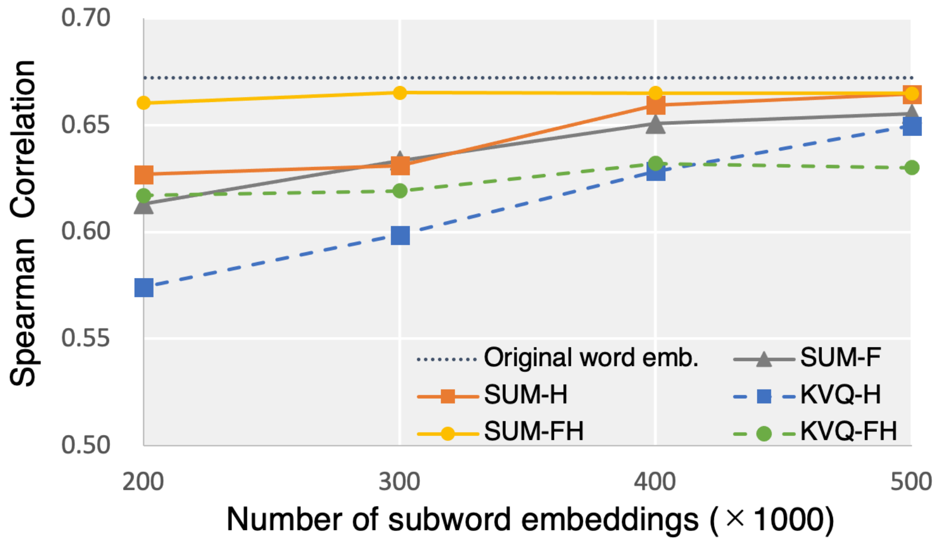


Figure 4.4 Overall performance/P curves for each memory-shared method. The x-axis represents the number of hash functions P; the y-axis represents the overall performance evaluated over WordSim and Analogy.



(a) `fastText.600B`



(b) `GloVe.840B`

Figure 4.5 Performance/model size curves for WordSim when using `fastText.600B` and `GloVe.840B` as the reconstruction target. The x-axis represents the number of subword embeddings. The y-axis represents the performance evaluated by the macro-average of Spearman's rho.

Table 4.5 Results of model shrinkage experiments by reconstructing the `fastText.600B` (before the slash) or `GloVe.840B` (after the slash) embeddings. Each dataset in WordSim was evaluated by Spearman's rho. "Macro" represents the macro-average of Spearman's rho over all WordSim datasets.

method	F	H	P	MEN	MC	MTurk	RW	R&G	SCWS	Slex	WSR	WSS	Macro
`Org.`	-	-	-	.815/.805	.850/.788	.735/.693	.572/.462	.871/.769	.684/.632	.471/.408	.640/.688	.835/.803	.719/.672
`SUM-F`	0.5M	-	-	.768/.784	.829/.750	.746/.684	.566/**.508**	.817/.696	.668/.639	.402/.421	.578/.648	.806/.767	.687/.655
`SUM-H`	-	0.5M	5	.751/.808	**.852**/.755	.711/**.720**	.528/.474	.837/.735	.659/.623	.375/.412	.553/**.675**	.804/**.777**	.674/.664
`KVQ-H`	-	0.5M	6	.802/.785	.842/**.775**	.742/.675	.548/.473	.828/**.767**	.680/.606	**.442**/.384	.614/.605	.811/.776	.701/.650
`SUM-FH`	1.0M	0.5M	4	.774/**.812**	.789/.741	.747/.693	.566/.504	.811/.740	.659/**.644**	.397/**.422**	.585/.652	.805/.775	.682/**.665**
`KVQ-FH`	1.0M	0.5M	4	**.806**/.785	.840/.722	**.750**/.674	**.573**/.471	**.854**/.720	**.689**/.613	.435/.398	**.634**/.551	**.829**/.735	**.712**/.630
`SUM-F`	0.2M	-	-	.731/.726	.809/.703	.710/.659	.526/.484	.777/.629	.642/.613	.369/.387	.482/.577	.762/.740	.645/.613
`SUM-H`	-	0.2M	5	.677/.740	.783/**.826**	.679/.658	.480/.441	.735/.725	.614/.594	.312/.352	.405/.555	.760/.750	.605/.627
`KVQ-H`	-	0.2M	6	.736/.702	**.839**/.761	.679/.572	.498/.430	.814/.716	.658/.560	.389/.331	.501/.450	.752/.645	.652/.574
`SUM-FH`	1.0M	0.2M	4	.720/**.780**	.782/.796	**.728/.716**	.518/**.504**	.782/.705	.635/**.631**	.336/**.392**	.487/**.635**	.778/**.784**	.641/**.660**
`KVQ-FH`	1.0M	0.2M	4	**.763**/.738	.820/.783	**.728**/.614	**.540**/.466	**.816/.759**	**.664**/.584	**.386**/.340	**.545**/.564	**.796**/.706	**.673**/.617

4.5.3 Downstream Tasks

For the evaluation of downstream tasks, we use the CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003) for the NER experiment and the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) for the TE experiment.

We compare the performance of our reconstruction models obtained in the model shrinkage experiments (Section 4.5.1) with the performance of original word embeddings, namely, `fastText.600B` and `GloVe.840B`. For our reconstruction models, we calculate the embeddings of all the words in the datasets; thus, there are no OOV words when using our methods.

We use AllenNLP[9] to train the base NER and TE models. Specifically, in the NER experiments, the base NER model consists of a word embedding layer, character embedding with CNN layer, two bidirectional LSTM layers, and a conditional random field layer. In the TE experiments, the base TE model is the ESIM sequence model (Chen et al., 2017) which is composed of a biLSTM to encode the premise and hypothesis, followed by a matrix attention layer, a local inference layer, another biLSTM inference composition layer, a pooling layer and a final output layer. Table 4.9 shows the hyper-parameters of model and training settings, respectively, whose values are obtained from the recommended (default) values of each task implemented in the AllenNLP tool.

We also introduce one hyper-parameter K to rescale the embeddings (i.e., all the elements in the embeddings were multiplied by K) because we learned that the rescaling might significantly affect the overall performance of downstream tasks in certain situations. We search K from $[1, 5, 10, 20, 50, 100]$ on the validation set of each dataset.

4.6 Results

4.6.1 Model Shrinkage

Fig. 4.4 shows the results of the preliminary experiments, namely, the overall performance/P curves. According to the results, we obtain the best overall performance at $P = 5$ for `SUM-H`, $P = 6$ for `KVQ-H`, $P = 4$ for `SUM-FH` and $P = 4$ for `KVQ-FH`. We use the selected Ps for our methods in all the remaining experiments.

Fig. 4.5 shows the performance/model size (or performance/number of embedding vectors) curves for WordSim. Table 4.4 shows the statistics of each setting. Table 4.5 shows the detailed result for each dataset. These results demonstrate that

[9]https://allennlp.org/

A-i) when reconstructing `fastText.600B`, we observed a consistent tendency that the methods using KVQ obtained a better performance than the methods using SUM;

A-ii) when reconstructing `GloVe.840B`, we observed that KVQ showed no improvement in performance compared with SUM; and

A-iii) for some datasets, we observed that our reconstruction methods achieved the performance of the original word embeddings such as `fastText.600B` or `GloVe.840B` when $H = 0.5\text{M}$.

Fig. 4.6 shows the performance/model size (or performance/number of embedding vectors) curves for Analogy. Table 4.6 shows the detailed result of the Analogy experiments for each dataset. These results demonstrate that

A-iv) the methods using KVQ obtained better results than the methods using SUM in most cases, and

A-v) notably, `KVQ-H` significantly outperformed `SUM-F` and `SUM-FH` by more than 10 points in terms of the micro-average accuracy of all Analogy datasets when $H = 0.5\text{M}$.

Based on result A-iii, our methods with $H = 0.5\text{M}$ successfully reduced the model size nearly fourfold compared with the original word embeddings while maintaining the original performance.

4.6.2 Experiments of OOV Word Embeddings

Table 4.7 shows the results of the synthetic OOV word experiments. The results indicate that

B-i) the performance of the `Random` baseline was nearly zero on the WordSim and exactly zero on the Analogy datasets; and

B-ii) the performances of `KVQ-FH` were significantly improved compared with that of `Random`, by 38–52 points on WordSim and by 7–28 points on Analogy.

Table 4.8 shows the results of nonsynthetic OOV word experiments to compare our method with previous studies.

We observed that

B-iii) `KVQ-FH` outperformed `BoS`, the previous state-of-the-art method, with substantial improvements by 6 points; and

Table 4.6 Results of model shrinkage experiments by reconstructing the `fastText.600B` (before the slash) or `GloVe.840B` (after the slash) embeddings. Each dataset in Analogy was evaluated by accuracy. "Micro" represents the micro-average of accuracy over all Analogy datasets.

method	F	H	P	GL	MSYN	Micro
`Org.`	-	-	-	84.9 / 82.3	87.8 / 80.7	85.6 / 81.9
`SUM-F`	0.5M	-	-	53.3 / 34.6	71.2 / 45.4	58.0 / 37.5
`SUM-H`	-	0.5M	5	65.1 / 63.8	83.6 / **71.9**	70.0 / 66.0
`KVQ-H`	-	0.5M	6	**77.1** / **68.3**	**86.6** / 69.6	**79.6** / **68.6**
`SUM-FH`	1.0M	0.5M	4	58.3 / 50.5	76.5 / 57.4	63.1 / 52.3
`KVQ-FH`	1.0M	0.5M	4	73.0 / 66.0	83.1 / 67.8	75.7 / 66.5
`SUM-F`	0.2M	-	-	40.3 / 26.1	66.2 / 42.1	47.2 / 30.3
`SUM-H`	-	0.2M	5	42.2 / 38.8	70.8 / 56.5	49.8 / 43.4
`KVQ-H`	-	0.2M	6	**48.1** / 40.8	**74.8** / 59.3	**55.2** / 45.7
`SUM-FH`	1.0M	0.2M	4	39.9 / 32.5	65.7 / 48.7	46.7 / 36.8
`KVQ-FH`	1.0M	0.2M	4	47.8 / **42.6**	72.2 / **59.5**	54.2 / **47.1**

Table 4.7 Results of (synthetic) OOV word experiments on WordSim and Analogy by reconstructing the `fastText.600B` (before the slash) or `GloVe.840B` (after the slash) embeddings. Performance was evaluated by Spearman's rho or accuracy.

method	F	H	P	WordSim	Analogy
`Random`	-	-	-	.110 / .036	0.0 / 0.0
`KVQ-FH`	1.0M	0.5M	4	.629 / .411	26.3 / 7.1
`KVQ-FH`	1.0M	0.2M	4	.620 / .465	27.8 / 8.3

B-iv) `KVQ-FH` achieved the highest performance in this comparison.

The result B-i means that we observe no correlation between the `Random` and human-annotated scores, which is a reasonable observation because OOV words are always contained in the evaluation WordSim/Analogy data, that is, a pair of two words/a list of four words. `Random` provides random vectors for the OOV words, thus we expect the problems to be unsolvable. Despite this difficult situation, `KVQ-FH` has large gains in performance from `Random` (B-ii), indicating that `KVQ-FH` successfully predicted the OOV word embeddings.

In the nonsynthetic experiments, the previous studies had little improvement from `Random`. However, `KVQ-FH` has substantially improved the performance compared with the ones of the `Random` and the previous methods (B-iii, B-iv).

Table 4.8 Results of OOV experiments on the Stanford Rare Word dataset. * indicates the values reported by Zhao et al. (Zhao et al., 2018). Note that `fastText` learned subword embeddings from an English Wikipedia dump because this method is not a reconstruction method.

method	F	H	P	$\|W\|$	$\|S\|$	RW
`Random`	-	-	-	0.16M	-	.452
`MIMICK` (Pinter et al., 2017)	-	-	-	0.16M	<1K	.201
`BoS` (Zhao et al., 2018)	-	-	-	0.16M	0.53M	.46*
`KVQ-FH`	0.50M	0.04M	4	0.16M	0.50M	**.521**
`fastText` (Bojanowski et al., 2017)	-	-	-	0.16M	0.53M	.48*

Table 4.9 Model and training settings of the NER and TE experiments.

		NER	TE
Model setting	Word embedding layer (dimension)	300	
	Character embedding layer (dimension)	16	—
	w/ CNN (kernel size)	3	—
	(filter size)	128	—
	Hidden state (dimension)	200	300
Training setting	Optimizer	Adam	
	Mini-batch size	64	32
	Dropout rate	0.5	

4.6.3 Evaluation on Downstream Tasks

To investigate the effectiveness of our reconstructed embeddings in downstream tasks, we evaluated them in the NER and TE tasks. Tables 4.10 and 4.11 show a comparison between the original (large) embeddings and our reconstructed (small) embeddings. We observed that

C-i) when $H = 0.5$M, the performances of our reconstructed embeddings are equivalent to or even better than those of the original embeddings; and

C-ii) when $H = 0.2$M, the performances of our reconstructed embeddings are comparable to them.

We could have been surprised by such comparable (or possibly improved) results for `KVQ-FH` because the model sizes of `KVQ-FH` were relatively very small compared with the original embeddings. However, we consider that this observation might be reasonable because our method additionally provided the embeddings of OOV words that cannot be handled by the original embeddings. Actually, we confirmed that the OOV rates for the `fastText.600B` vocabulary are 12.8% in NER and 11.9% in TE tasks. For the `GloVe.840B` vocabulary, the OOV rates are 11.4% in NER and 11.9% in TE tasks.

Table 4.10 Results of the NER experiments on the CoNLL-2003 dataset.

method	F	H	P	K	size (GB)	F1
`fastText.600B`	-	-	-	20	2.26GB	90.3
`KVQ-FH`	1.0M	0.5M	4	100	0.59GB	89.9
`KVQ-FH`	1.0M	0.2M	4	50	0.23GB	89.4
`GloVe.840B`	-	-	-	10	2.48GB	90.8
`KVQ-FH`	1.0M	0.5M	4	100	0.59GB	90.9
`KVQ-FH`	1.0M	0.2M	4	100	0.23GB	90.4

Table 4.11 Results of the TE experiments on the SNLI dataset.

method	F	H	P	K	size (GB)	Acc
`fastText.600B`	-	-	-	10	2.23GB	87.8
`KVQ-FH`	1.0M	0.5M	4	10	0.59GB	87.5
`KVQ-FH`	1.0M	0.2M	4	20	0.23GB	87.6
`GloVe.840B`	-	-	-	1	2.45GB	88.3
`KVQ-FH`	1.0M	0.5M	4	10	0.59GB	87.6
`KVQ-FH`	1.0M	0.2M	4	10	0.23GB	87.3

4.7 Analysis

To deeply understand our models, we conduct the following four analyses:

- Calculation speed analysis
- Syntactic and semantic Analogy tests
- An investigation of the impact of hyper-parameter selection
- An investigation of the distribution of `FNV` hash function

4.7.1 Calculation Speed

Models should quickly calculate word embeddings for real-world applications that require immediacy, such as a real-time translation system. We investigate the difference in calculation speed between `SUM-FH` and `KVQ-FH`. To achieve this objective, we calculate the time required to compute word embeddings from subword embeddings by using each operation (Table 4.12). The speed is slower when using a larger P because preparing the indexes of subword vectors takes more time in using multiple hash functions. When $P = 1$, `SUM-FH` and `KVQ-FH` took 3.6×10^{-4} and 7.5×10^{-4} seconds per word, respectively, namely, KVQ took 2.1 times longer than SUM. However, the calculation speed per word is sufficiently high, for example, at most, `KVQ-FH` with $P = 10$ took 4.4×10^{-3} seconds per word. This can

Table 4.12 Calculation speed (in seconds) of SUM-FH and KVQ-FH. P is the number of hash functions used in our memory sharing method (mentioned in Section 3.3.2).

P	SUM-FH	KVQ-FH
1	3.6×10^{-4}	7.5×10^{-4}
2	6.2×10^{-4}	1.4×10^{-3}
3	9.0×10^{-4}	2.1×10^{-3}
5	1.4×10^{-3}	2.9×10^{-3}
10	2.6×10^{-3}	4.4×10^{-3}

be negligible in real applications because other operations such as calculating deep neural networks may take much longer.

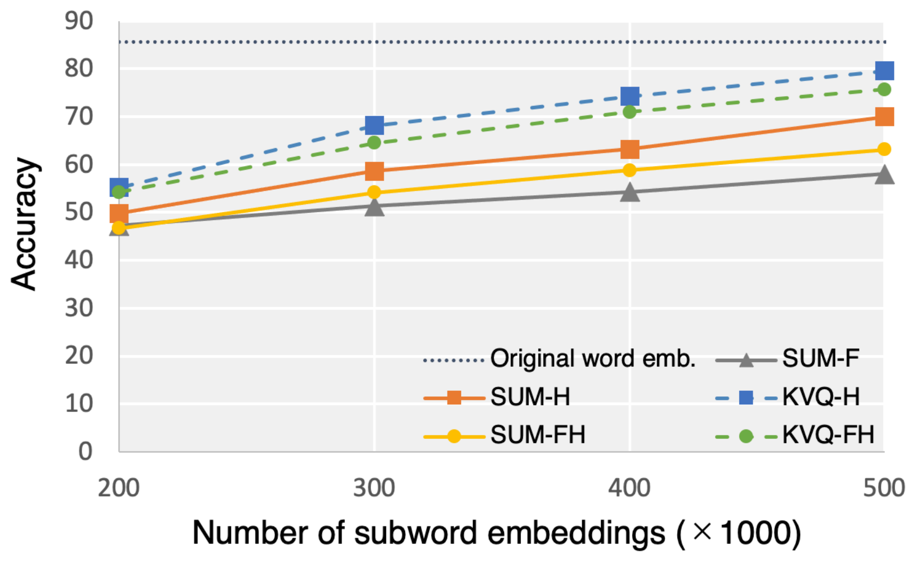
4.7.2 Syntactic and Semantic Tests on Analogy

The questions in Google's analogy dataset (Mikolov et al., 2013a) can be split into two types, namely, semantic and syntactic test sets. We also used these categorized test sets to examine the model's behavior in the Analogy evaluation. The semantic tests consist of questions assessing whether models can catch semantic relationships between words, e.g., "France" is to "Paris" as "Germany" is to "Berlin". On the other hand, the syntactic tests consist of questions about syntactic relationships, e.g., "small" is to "smallest" as "big" is to "biggest". In this analysis, we used models with $H = 0.5$M trained by reconstructing `fastText.600B`. For KVQ models, we used KVQ-H with $P = 6$ and KVQ-FH with $P = 4$. Fig. 4.7 shows the model performances on the categorized Analogy datasets. Orange lines show the performances of SUM; green lines show the performances of KVQ. Although the SUM models and the KVQ models had comparable performances on the syntactic test sets, the SUM models had poor performances on the semantic test sets. We posit that the syntactic tests can be answered by using partial information from word's suffixes, but the semantic tests require the models to be more expressive.

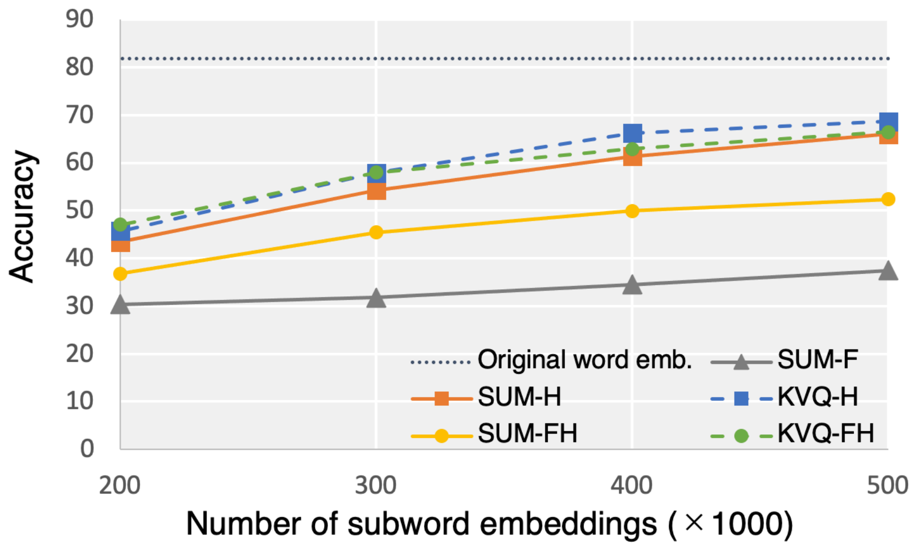
4.7.3 Impact of Hyper-parameter Selection

We investigated the impact of the hyper-parameter selection of each component in our method. Specifically, we evaluated the effects of changing the hyper-parameters of (1) F in the combination methods (Section 4.4.3) and (2) Z appearing in Eq. 4.16. In all the experiments in these investigations, we used the KVQ-FH ($H = 0.2$M or $H = 0.5$M and $P = 4$) with `fastText.600B` as the reconstruction target embeddings. We evaluated the performances on the WordSim and Analogy datasets as described in Table 4.2, similarly to in the experiments in Section 4.5.1.

Fig. 4.8 shows the performance/F value curves for WordSim and Analogy. The performance was improved by using a larger value of F on Analogy datasets, and on WordSim datasets, the performance worsened. We recognize that the models may be more expressive, especially for infrequent words, by taking advantage of many infrequent subword vectors, i.e., setting a large value into F, contributing to the improvement of the performance on Analogy. However, taking advantage of too many subwords and sharing their vectors may result in negative effects especially for the reconstruction of frequent word vectors, causing the performance degradation on WordSim since most of words in WordSim datasets are frequent words.

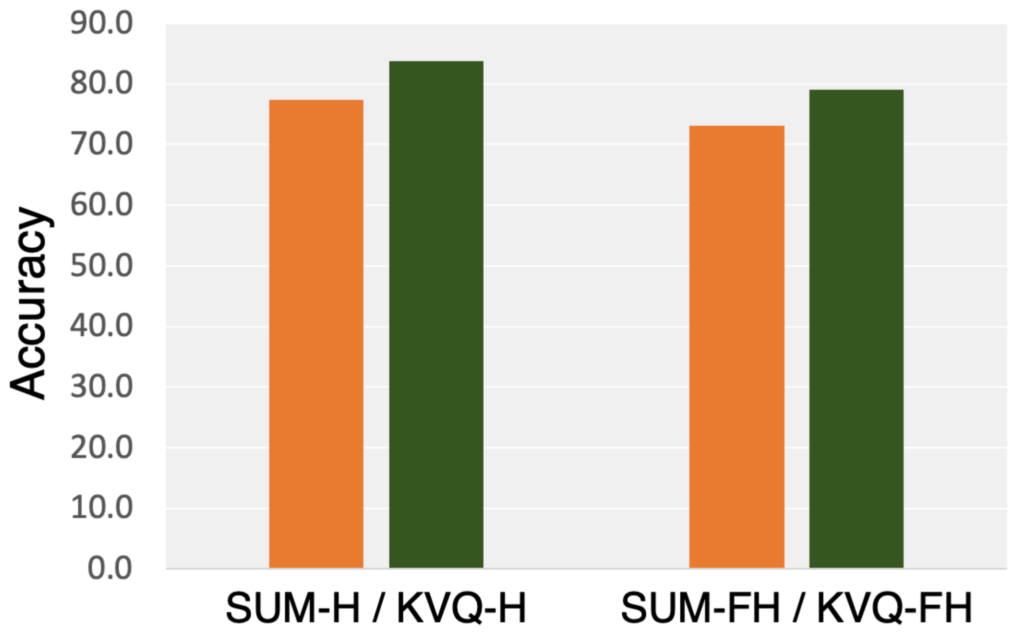


(a) `fastText.600B`

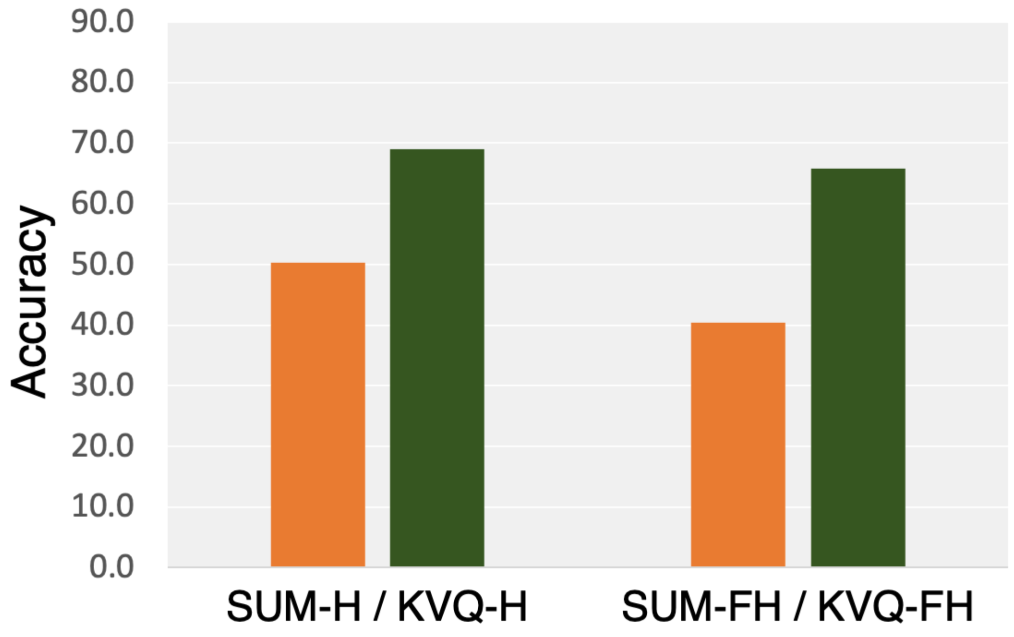


(b) `GloVe.840B`

Figure 4.6 Performance/model size curves for Analogy when using `fastText.600B` and `GloVe.840B` as the reconstruction target. The x-axis represents the number of subword embeddings. The y-axis represents the performance evaluated by the micro-average accuracy.

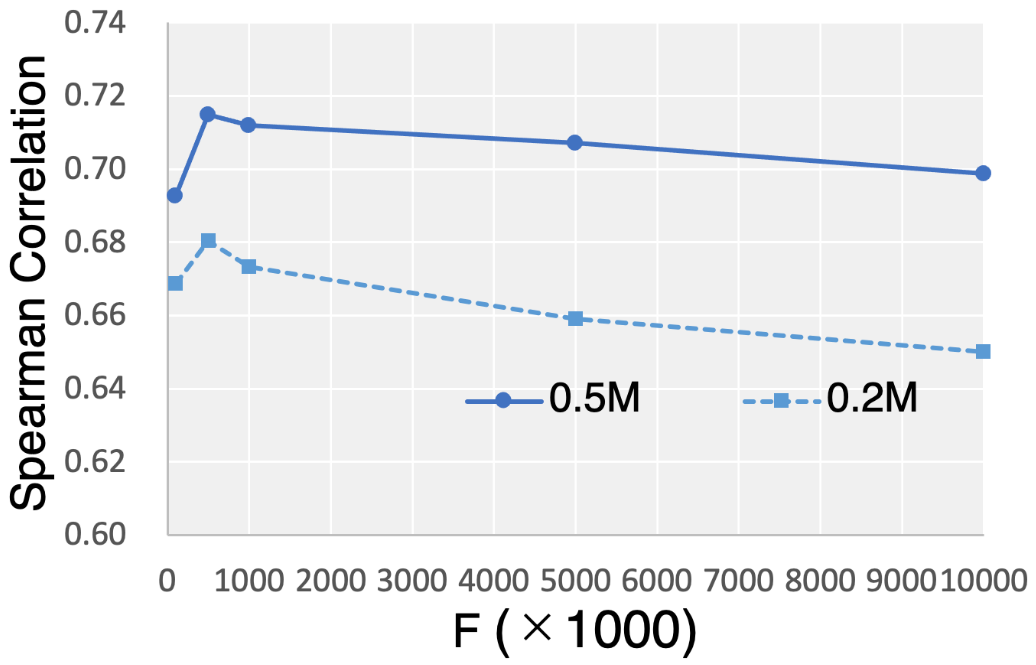


(a) Syntactic tests

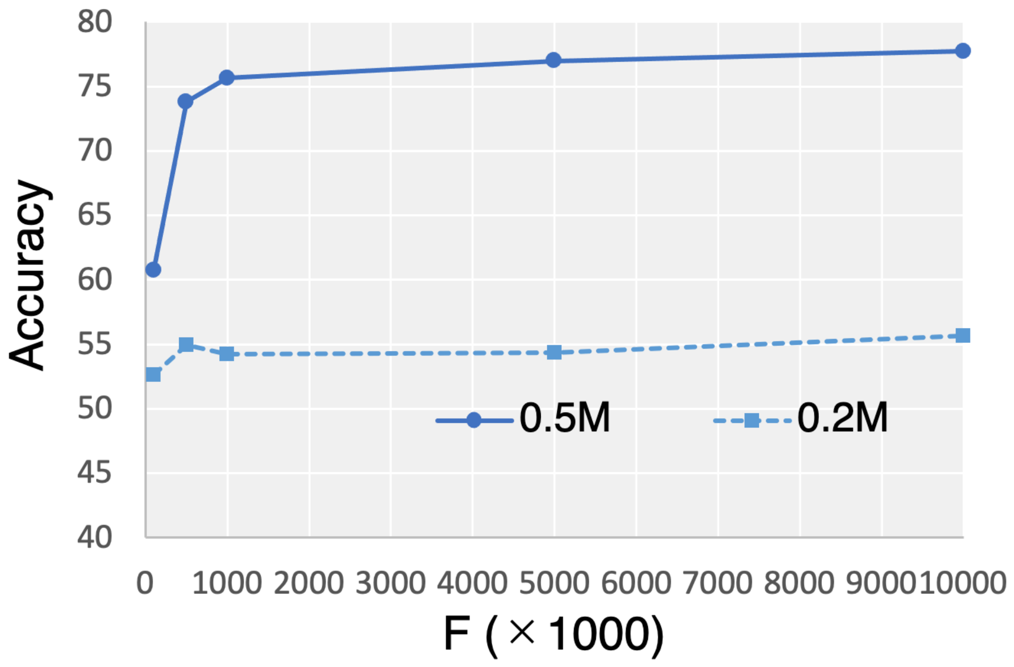


(b) Semantic tests

Figure 4.7 Performances on the syntactic and semantic tests of Analogy datasets. Orange lines show the performances of SUM; green lines show the performances of KVQ.

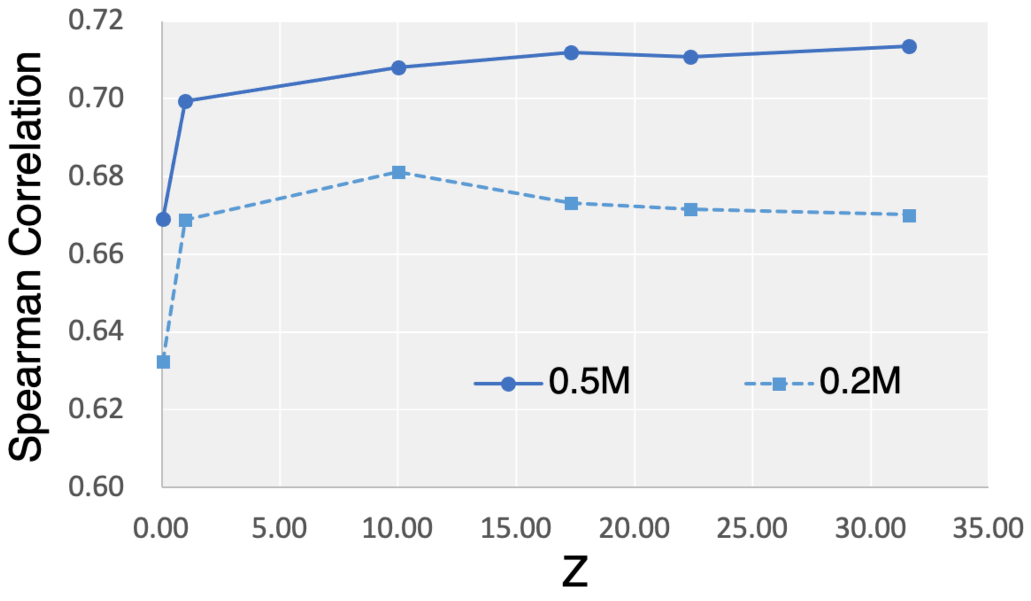


(a) WordSim

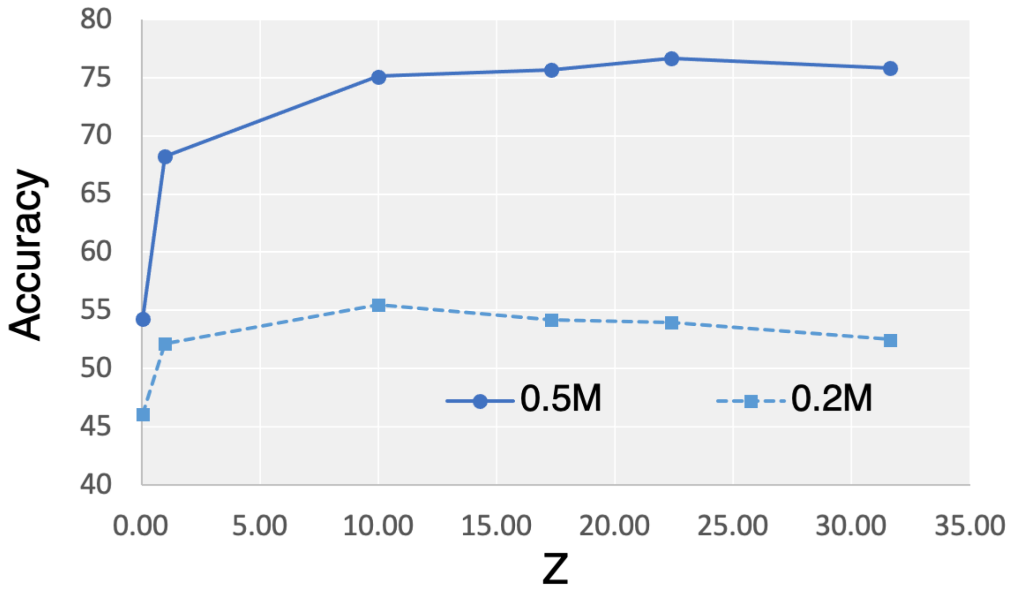


(b) Analogy

Figure 4.8 Performances on WordSim and Analogy tests. Blue lines show the performances when $H = 0.5$M; dashed lines show the performances when $H = 0.2$M.



(a) WordSim



(b) Analogy

Figure 4.9 Performances on WordSim and Analogy tests. Blue lines show the performances when $H = 0.5\text{M}$; dashed lines show the performances when $H = 0.2\text{M}$.

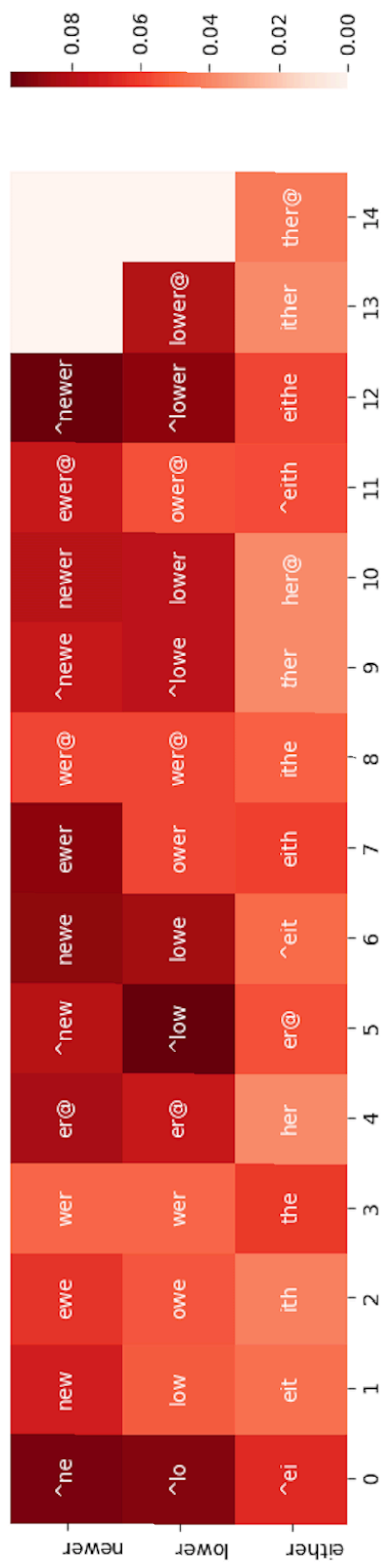
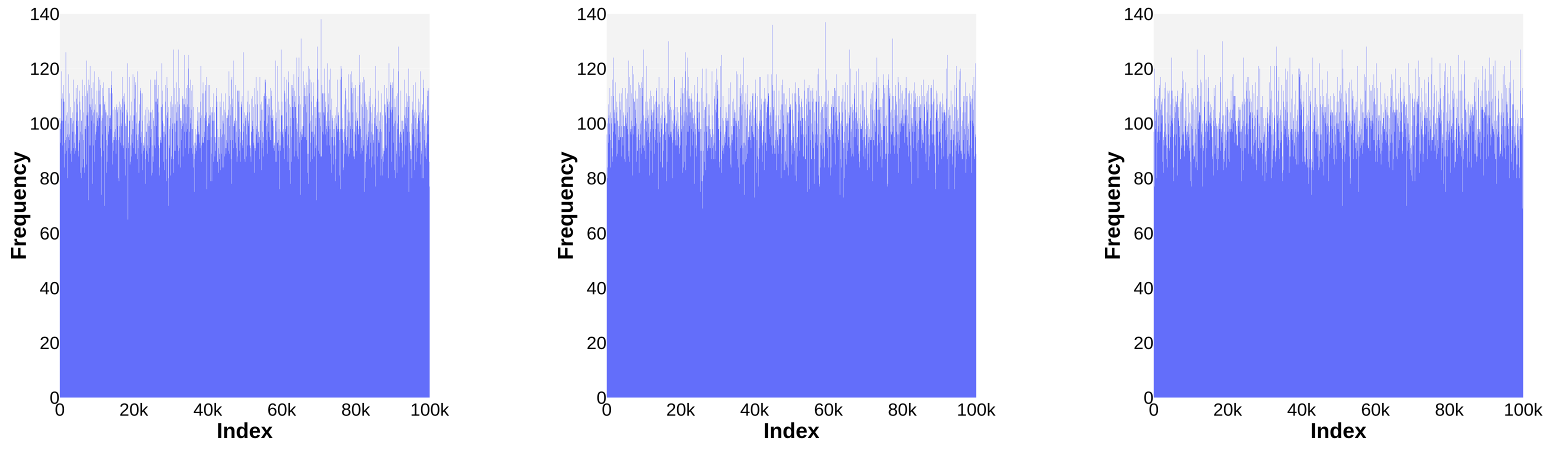


Figure 4.10 Attention distribution of our KVQ model. Here we used “^” and “@” as special characters that represent the beginning and end of a word, respectively. Subwords in boxes are corresponding subwords obtained from a word at the left side.



(a) `Numpy.random.randint` (b) Random strings (c) Subwords

Figure 4.11 Histograms of (a) the values from Numpy.random.randint, (b) the `FNV` hashes when inputting random strings, and (c) the `FNV` hashes when inputting subwords from a vocabulary of `fastText.600B`.

Fig. 4.9 shows the performance/Z value curves for WordSim and Analogy. We found that the KVQ model had the best performance on both WordSim and Analogy when we set Z to approximately $\sqrt{D}$. Notably, this value is relatively larger than that used in `Transformer` (Vaswani et al., 2017), that is, $Z = 1/\sqrt{D}$. We then checked actual attention distributions in `KVQ-FH` to investigate the tendency. Fig. 4.10 shows typical examples by some picked words. We observed that attention distributions tend to be a flat distribution rather than a peaky distribution, but never match a near-uniform distribution. This implies that the attention mechanism works as the weighting factor to assign the importance to subwords. However, this is also worth noting here that determining the importance of subwords seems not an easy task. This might be the reason why Z tends to require a relatively large value. If we select a smaller Z, the distribution may become closer to the uniform distribution, and the performance would decrease since the uniform distribution is essentially identical to SUM-FH. Moreover, we observed that different attention values were assigned to the identical subwords when their "context" subwords differed; thus, the KVQ models successfully captured "context-dependent" weighting for each subword.

4.7.4 Distribution of FNV Hash

For the mapping function $\eta_H(\cdot)$ in Eq. 4.11, we used a `FNV` hash function, following a previous study (Bojanowski et al., 2017). `FNV` is a lightweight hash function that takes a string and returns a random integer from a specific distribution. We expected the distribution to be uniform, however, there is no theoretical proof of it as far as we know.

To clarify the empirical distribution of `FNV`, we plotted histograms of the `FNV` hash values and compared them with that of the standard random function, such as '`Numpy.random.randint`' implemented as a Python library. Figure 4.11 shows the results of this experiment. Figure 4.11a is the histogram of the '`Numpy.random.randint`' values sampled 10 million times from the range of $[0, H-1]$. H corresponds to the hash size of our method. Here we use $H = 100000$. Then, Figure 4.11b and 4.11c are the histograms of the `FNV` hash values divided by H based on the randomly generated strings 10 million times or 10 million subwords obtained from a vocabulary of `fastText.600B` embeddings, respectively. Additionally, we conducted an F-test to check the equality of variances between Figure 4.11b and 4.11c, and confirmed the p-values was 0.734. The 95% confidence intervals for variances were $99.8 <= \sigma^2 <= 101.6$ for Figure 1(a), $99.6 <= \sigma^2 <= 101.4$ for Figure 1(c), respectively. The mean value was 100 for both. These indicate the distribution of `FNV` hash values is quite similar or almost equal to that of the standard random function. We can conclude that the `FNV` hash function we used for our experiments was random enough as well as a standard random function implemented as a well-used library.

4.8 Conclusion

We have discussed and investigated an approach that reconstructs subword-based word embeddings in a reduced memory space. We have demonstrated that memory-shared embeddings with the KVQ self-attention operation significantly outperformed the conventional summation-based approaches, such as `BoS`. Moreover, our best setting successfully reduced the number of embedding vectors to approximately 10 times smaller than that of the original word embeddings while maintaining an acceptable performance loss on downstream tasks. We have also confirmed the effectiveness of our approach in its the applicability of OOV words. We believe that our reconstructed subword-based word embeddings can be better alternatives to `fastText.600B` and `GloVe.840B` because they require less memory requirement and have high applicability of OOV words.[10]

[10] Our code and reconstructed subword-based word embeddings trained from `GloVe.840B` and `fastText.600B` are available at https://github.com/losyer/compact_reconstruction

Chapter 5

Examining the Effect of Whitening on Static and Contextualized Word Embeddings

5.1 Introduction

Static word embeddings (SWE) (Mikolov et al., 2013c; Pennington et al., 2014b) and contextualized word embeddings (CWE) (Devlin et al., 2019; Peters et al., 2018b; Raffel et al., 2020, i.a.) are the foundation of modern natural language processing systems. However, while the aim of creating such embeddings is to provide accurate representations of word, phrase, and sentence meaning, they also reflect and sometimes amplify biases inherent in the training data, such as gender bias (Zhao et al., 2019), social bias (Kaneko and Bollegala, 2022), and word frequency bias (Gong et al., 2018). For SWE, prior research has demonstrated that the embedding space exhibits a spatial frequency bias; namely, frequent words tend to concentrate along a particular direction (Mu and Viswanath, 2018). Generally, this *anisotropy*, i.e., the non-uniform angular distribution of word vectors, is undesirable because it leads to inefficient use of the embedding space. Furthermore, frequency-based anisotropy causes frequent words to be represented by similar vectors simply by virtue of their high frequency, although their meaning may not be similar.

Aiming to reduce the negative impact of anisotropy, several *isotropization* methods have been proposed. These methods make embeddings more *isotropic*, i.e., transform embedding vectors so that they have a more uniform angular distribution. Isotropization methods can be divided into supervised debiasing methods and unsupervised post-processing methods.

The primary goal of supervised debiasing methods is to remove biases with respect to specific categories, such as gender, nationality, and word frequency. If the bias manifests itself as an uneven distribution of word embeddings, then debiasing results in a more isotropic embedding space. A representative example of such an approach is the adversarial removal of protected social variables proposed by Zhang et al. (2018). In contrast to supervised debiasing methods, unsupervised post-processing methods aim to improve word embeddings without relying on word meaning or associated categories.

The focus of this study is the arguably simplest and most common unsupervised isotropization method, namely *whitening*. Informally, whitening is a linear operation that transforms a set of spatially correlated (and therefore anisotropic) vectors into a set of uncorrelated (isotropic) vectors (see Section 5.2.2 for a formal definition). Although whitening is a standard data transformation technique, applications to NLP and CWE, in particular, have appeared only recently (Huang et al., 2021; Su et al., 2021). These applications have demonstrated that whitening performs better than other isotropization methods for CWE. However, a major disadvantage of whitening and other unsupervised post-processing methods is that their impact on various forms of bias and other semantic properties of embeddings is not yet understood, although understanding bias in SWE and CWE is a prerequisite for their ethical use in real-world applications. In this paper, we present an initial analysis of the semantic impact of unsupervised post-processing. In particular, we analyze changes in the frequency bias when applying the whitening transformation to SWE and CWE.

Our preliminary analysis indicates that the effect of whitening partially includes the effect of frequency debiasing. Our research question is thus whether the effect of whitening consists of frequency debiasing only. To increase the granularity of the effect of whitening, we employ a method whose effect is frequency debiasing only; specifically, we propose a reconstruction-based frequency debiasing (RFD), which focuses only on removing frequency bias in embeddings. We then compare the behavior of whitening with that of RFD. Our experimental results indicate that whitening removes word frequency bias in SWE as well as biases other than word frequency bias in CWE.

5.2 Background

5.2.1 Anisotropy in Static and Contextualized Word Embeddings

Mu and Viswanath (2018) reported an anisotropy problem that leads to reduce expressiveness of SWE. Ethayarajh (2019) reported that anisotropy also existed in CWE. Accordingly, there has been much discussion about the causes of anisotropy in embeddings. Mu and

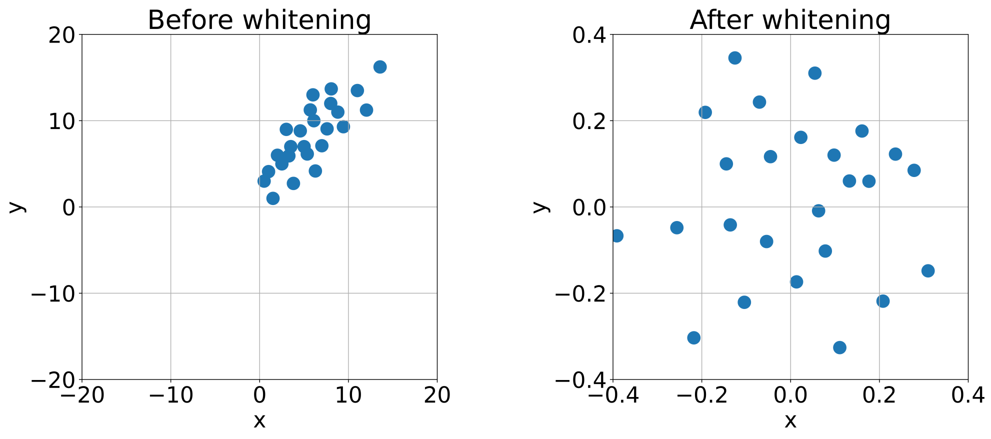


Figure 5.1 Examples of plots before and after applying whitening to a set of two-dimensional vectors. Before applying whitening, x and y are correlated, whereas after applying whitening, they are no longer correlated.

Viswanath (2018) reported that word frequency information is embedded in the first and second principal components of SWE, and is the cause of anisotropy in SWE. In CWE, frequency bias is the most common issue. Li et al. (2020) empirically demonstrated that there is a frequency bias in the vectors of the word embedding layer in BERT Devlin et al. (2019). Specifically, they demonstrated that vectors of frequent words are embedded closer to the origin, while infrequent words are embedded farther from the origin. Moreover, they showed that vectors of high-frequency words are densely embedded, while vectors of low-frequency words sparsely dispersed. Liang et al. (2021) also reported that there is a correlation between the logarithm of word count and the norm/average cosine similarity of word vectors. In addition to frequency bias, outlier dimensions in CWE have also recently received attention. Luo et al. (2021) and Kovaleva et al. (2021) identified dimensions in the embeddings of BERT and RoBERTa Liu et al. (2019) that were significantly higher than other dimensions, suggesting that they were the cause of anisotropy in the embeddings.

5.2.2 Isotropization via the Whitening Transformation

Whitening is a linear transformation that transforms a set of vectors into a new set where the covariance matrix is the identity matrix. The fact that the covariance matrix is an identity matrix signifies that the transformation makes each dimension uncorrelated (*uncorrelation*) and sets the variance to 1 (*variance flattening*). Through the transformation, the resulting whitened embeddings become more isotropic (Figure 5.1). Let $X \in \mathbb{R}^{N \times d}$ be a set of vectors such as word embeddings. The embeddings transformed by whitening are defined as follows:

$$X' = (X - m)U\sqrt{S^{-1}}, \tag{5.1}$$

where m is the mean vector of the vectors in X, and U and S are given by singular value decomposition of the covariance matrix Σ of X:

$$\Sigma = USU^T. \tag{5.2}$$

Whitening is commonly used in machine learning to reduce bias in the training data, and it has been applied to a set of feature vectors as a feature preprocessing method (Coates et al., 2011; Ranzato et al., 2010). It has been reported that reducing bias helps deep learning models learn high-quality representation and speeds up model convergence. Since whitening is a general-purpose algorithm that can be applied to a set of vectors, it can also be applied to sentence vectors obtained by CWE. Huang et al. (2021) applied whitening to CWE to address the anisotropy problem and demonstrated that it improved the performance of the CWE. Whitening is mathematically well defined; however, analysis has not yet been performed to clarify what information in SWE and CWE is processed and how the information is transformed by whitening. In this study, we aim to clarify the mechanism of whitening (i.e., uncorrelation and variance flatting) in SWE and CWE.

5.2.3 Other Isotropization Methods

In addition to whitening, several other methods to address anisotropy have been proposed. One class of isotropization methods removes the principal components of embeddings. Mu and Viswanath (2018) suggested that there is a word frequency bias in SWE, and reported that the bias negatively affects task performance. Specifically, they observed a frequency bias in the first and second principal components of GloVe (Pennington et al., 2014b) and Word2Vec (Mikolov et al., 2013c) embeddings, and proposed a method to remove the top-D principal components, denoted the `RPC` method. They found that the `RPC` method improves the performance of tasks and reduces the anisotropy of SWEs. Rajaee and Pilehvar (2021) proposed a cluster-based version of the `RPC` method, while Liang et al. (2021) proposed a weighted version.

Several mathematically-motivated methods have also been proposed to handle anisotropic embeddings. Li et al. (2020) proposed `BERT-flow`, which learns a flow function that projects embeddings obtained from BERT to a standard Gaussian latent space. Their theoretical motivation was that embeddings with a standard Gaussian distribution are a suf-

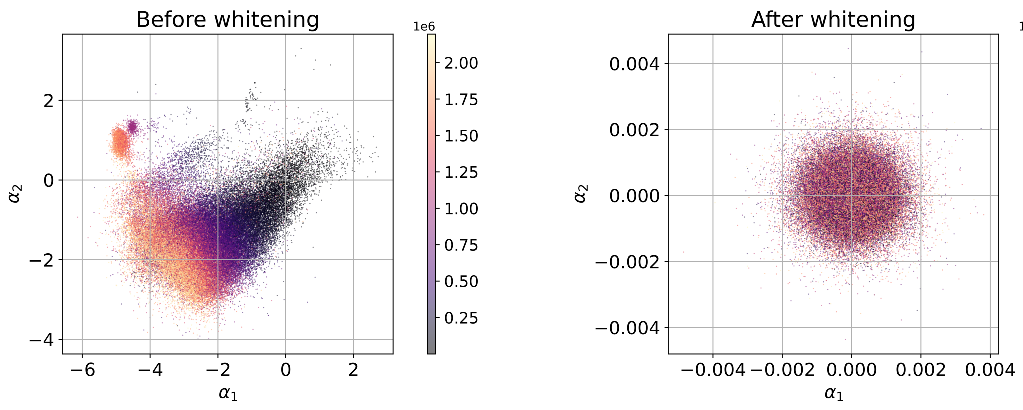


Figure 5.2 Plots of the first and second principal components (α_1 and α_2) before and after applying whitening to GloVe embeddings. Colors correspond to word frequency ranks. Black represents frequent words, while yellow represents infrequent words.

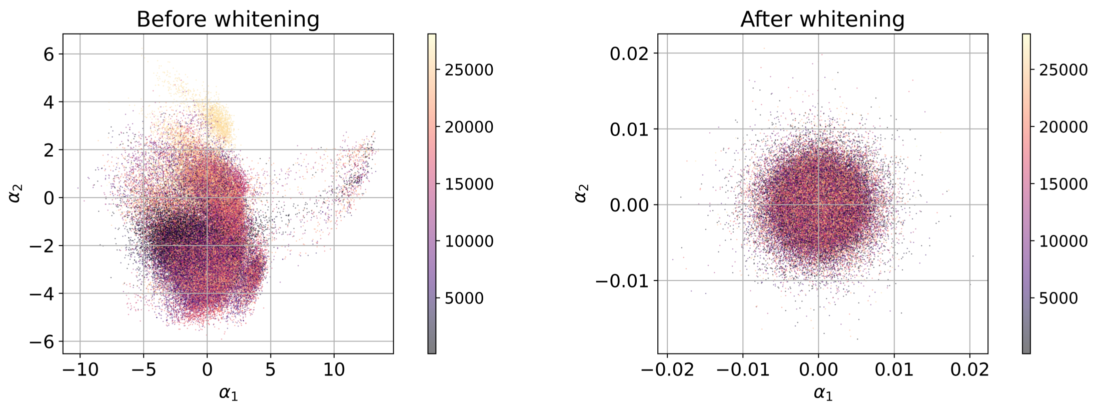


Figure 5.3 Plots of the first and second principal components (α_1 and α_2) before and after applying whitening to BERT embeddings. Colors correspond to word frequency ranks. Black represents frequent words, while yellow represents infrequent words.

ficient condition for isotropy and an efficiently embedded space, which they described as lacking holes.

5.3 Preliminaries

Following the work of Mu and Viswanath (2018), we performed an analysis to explore the effect of whitening in preliminary experiments. Figures 5.2 and 5.3 present plots of the first and second principal components of the embeddings before and after applying whitening to GloVe and BERT embeddings, respectively. Before applying whitening, we observed a frequency bias, i.e., a correlation between the principal components of word embeddings and their word frequencies in both GloVe and BERT embeddings. In particular, GloVe embeddings had a strong frequency bias, which is consistent with the reports from Mu and Viswanath (2018). However, after applying whitening, we found that there was no bias in the first and second principal components of the embeddings, which indicates that whitening has the effect of frequency debiasing.

Based on this analysis, we aim to clarify the effect of whitening. Our research question is whether whitening is equivalent to frequency debiasing or whether it has effects other than frequency debiasing. To address this question, we conduct an experiment in which we apply both whitening and a frequency debiasing method, which is introduced in Section 5.4, to the model at the same time. When both are applied to the model, if the effects of whitening and frequency debiasing are independent, then respective gains can be expected, while if there is an overlap between the two effects, then the improvements can be limited.

5.4 Frequency Debiasing Method

In this section, we introduce a frequency debiasing method SWE and CWE that focuses only on the effect of frequency debiasing without affecting the original quality of the embeddings. Gong et al. (2018) proposed a method of frequency debiasing for word embeddings using adversarial training. Given a certain task, such as text classification, their method optimizes each vector in the word embeddings layer with the debiasing loss simultaneously as learning. With these settings, the resulting word embeddings are task-specific representations. We aim to obtain general representations rather than task-specific representations; thus, we propose a reconstruction-based frequency debiasing (RFD) inspired by Gong et al. (2018). Like Gong et al. (2018), we assume that word embeddings are trained to fool a discriminator attempting to identify whether the words are rare or popular. Instead of a task-specific loss, we introduce a reconstruction loss that aims to preserve the pretrained representations such as embeddings from GloVe or BERT.

First, we present a reconstruction loss for SWE such as GloVe. Let W be a set of words in a vocabulary, $\boldsymbol{e}(w)$ be the pretrained fixed embeddings of word w and $\boldsymbol{v}(w; \theta^{\text{emb}})$ be the learnable embeddings of word w, where $\theta^{\text{emb}} \in \mathbb{R}^{d \times V}$ is the parameter matrix of the word embeddings. Here d is the number of dimensions of the embeddings and $V (= |W|)$ is the vocabulary size. The reconstruction loss for SWE is defined as follows:

$$L_{R_{\text{swe}}}(W; \theta^{\text{emb}}) = \sum_{w \in W} \left\| \boldsymbol{e}(w) - \boldsymbol{v}(w; \theta^{\text{emb}}) \right\|_2^2. \tag{5.3}$$

Regarding the discriminator part for SWE, we follow the settings used by Gong et al. (2018)'s settings. First, we divide the vocabulary W into two parts: W_{pop} and W_{rare}. Words in W_{pop} are the top-$t\%$ frequent words, while $W_{\text{rare}} = W \setminus W_{\text{pop}}$. Let f_{θ^D} represent a discriminator with parameters θ^D that takes word embeddings as input and returns a probability score indicating whether the word is rare or not. The loss of the discriminator for SWE is defined as follows:

$$L_{D_{\text{swe}}}(W; \theta^D, \theta^{\text{emb}}) = \frac{1}{|W_{\text{pop}}|} \sum_{w \in W_{\text{pop}}} \log f_{\theta^D}(\mathbf{v}(w; \theta^{\text{emb}})) + \frac{1}{|W_{\text{rare}}|} \sum_{w \in W_{\text{rare}}} \log(1 - f_{\theta^D}(\mathbf{v}(w; \theta^{\text{emb}}))). \tag{5.4}$$

Lastly, we optimize θ^{emb} and θ^D using the adversarial training procedure with the min-max objective as follows:

$$\arg\min_{\theta^{\text{emb}}} \arg\max_{\theta^D} L_{R_{\text{swe}}}(W; \theta^{\text{emb}}) - \lambda L_{D_{\text{swe}}}(W; \theta^D, \theta^{\text{emb}}), \tag{5.5}$$

where λ is a hyperparameter used as a weight coefficient. Following Gong et al. (2018), we alternate between optimizing the argmin objective for θ^{emb} and optimizing the argmax objective for θ^D.

For CWE, we prepare a training corpus C and optimize the embeddings obtained by encoding sentences from C. Let s represent a sentence from corpus C, W_s represent a set of words[1] in s, and L be a target set of layers in a CWE model such as BERT. Let $\mathbf{e}^l(w, s)$ be the pretrained l-th layer embeddings of word w obtained by encoding sentence s. Embeddings $\mathbf{v}^l(w, s; \theta^{\text{emb}})$ is similar to $\mathbf{e}^l(w, s)$ but embeddings from a new CWE model with learnable parameters θ^{emb}. The reconstruction loss and discriminator loss for CWE are defined as follows:

$$L_{R_{\text{cwe}}}(C; \theta^{\text{emb}}) = \sum_{s \in C} \sum_{w \in W_s} \sum_{l \in L} \left\| \mathbf{e}^l(w, s) - \mathbf{v}^l(w, s; \theta^{\text{emb}}) \right\|_2^2, \tag{5.6}$$

$$L_{D_{\text{cwe}}}(C; \theta^D, \theta^{\text{emb}}) = \sum_{s \in C} \sum_{w \in W_s} \sum_{l \in L} L'_{D_{\text{cwe}}}(w, l; \theta^D, \theta^{\text{emb}}), \tag{5.7}$$

[1]Note that we use "word" to refer to "token" for the sake of clarity even though words are separated into tokens in CWE models; for example, "interferometer" is separated into "inter", "##fer" and "##ometer".

$$L'_{D_{\text{cwe}}}(w, l; \theta^D, \theta^{\text{emb}}) = \frac{1}{|W_{s,\text{pop}}|} \sum_{w \in W_{s,\text{pop}}} \log f_{\theta^D}(\mathbf{v}^l(w, s; \theta^{\text{emb}})) + \frac{1}{|W_{s,\text{rare}}|} \sum_{w \in W_{s,\text{rare}}} \log(1 - f_{\theta^D}(\mathbf{v}^l(w, s; \theta^{\text{emb}})), \tag{5.8}$$

where $W_{s,\text{pop}} = W_s \cap W_{\text{pop}}$ and $W_{s,\text{rare}} = W_s \setminus W_{s,\text{pop}}$. The objective for CWE is defined as follows:

$$\underset{\theta^{\text{emb}}}{\arg\min}\,\underset{\theta^D}{\arg\max}\, L_{R_{\text{cwe}}}(C; \theta^{\text{emb}}) - \lambda L_{D_{\text{cwe}}}(C; \theta^D, \theta^{\text{emb}}). \tag{5.9}$$

5.5 Experiments

We conduct experiments with several models to investigate the effect of whitening. When applied simultaneously, if the performance of whitening and frequency debiasing is equivalent to that of a single applied model, this indicates that whitening has the same effect as frequency debiasing.

5.5.1 Settings

We use GloVe embeddings (Pennington et al., 2014b) as the SWE. Of the various types of GloVe embeddings, we use embeddings trained on the Common Crawl dataset containing 840 billion tokens.[2] The vocabulary size is $V = 2,196,016$. We use a BERT-base model (Devlin et al., 2019) from Huggingface Transformer Library (Wolf et al., 2020) as the CWE. The vocabulary size is $V = 28,996$.

We compare four distinct settings of SWE and CWE:

1. Vanilla models (`GloVe`, `BERT`): models that use raw embeddings without post-processing.

2. Whitening models (`GloVe-wh`, `BERT-wh`): models with whitening applied.

3. Frequency debiasing models (`GloVe-Fdeb`, `BERT-Fdeb`): models with frequency debiasing applied.

4. Frequency debiasing and whitening models (`GloVe-Fdeb-wh`, `BERT-Fdeb-wh`): models with frequency debiasing applied followed by whitening.

[2]https://nlp.stanford.edu/projects/glove/

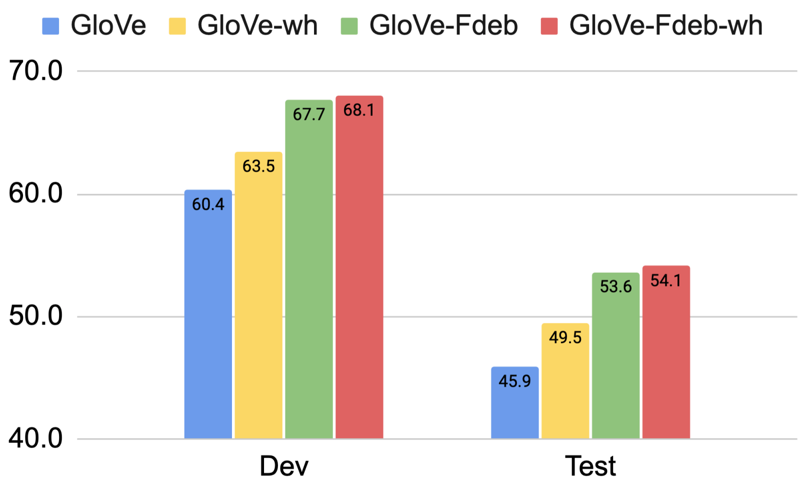


Figure 5.4 Performance of GloVe models on STS benchmark development and test set.

For frequency debiasing, we use `RFD`, which is introduced in Section 5.4. We use the sentences from the semantic textual similarity (STS) datasets as the training corpus for `RFD` for BERT. Following Gong et al. (2018), we use $\lambda = 0.02$. We also use $t = 10$ as the threshold for a set of frequent words. The target layer list L in equations **??** and **??** is $[1, 12]$. We fix the batch size to 128, and search the learning rate from $[1e{-}3, 5e{-}3, 1e{-}2]$, and the number of epochs from $[1, 3, 5, 10, 20, 30, 40, 50]$ in the development set.

5.5.2 Task

Dataset To evaluate the quality of the embeddings, we used the STS Benchmark dataset (Cer et al., 2017), which consists of sentence pairs with manually annotated scores as sentence similarities. The scores range from 0 to 5.

Evaluation Following previous studies (Huang et al., 2021; Reimers and Gurevych, 2019), we compute the Spearman rank correlation between the annotated ground truth scores and the similarities predicted by models. We calculate the cosine similarities of the sentence vectors as the similarities between sentences. For GloVe, the sentence vectors are calculated by averaging the vectors of all words. For BERT, following Huang et al. (2021), we average the vectors of words at the first and 12th hidden layers.

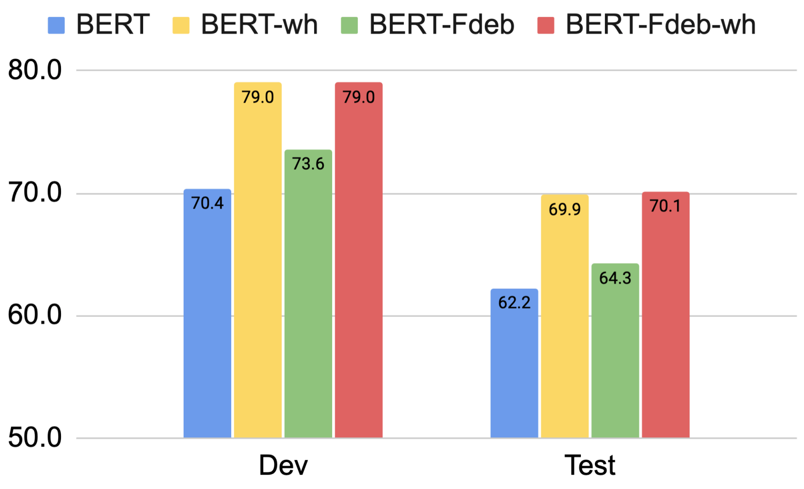


Figure 5.5 Performance of BERT models on STS benchmark development and test set.

5.5.3 Results

Figures 5.4 and 5.5 show the performance of the models on the STS Benchmarks development and test sets. The results are as follows:

(i) In both the GloVe and BERT experiments, whitening improved the performance. The performance of `GloVe-wh` was 3.6 points higher than that of `GloVe`, while the performance of `BERT-wh` was 7.7 points higher than that of `BERT` on the test set.

(ii) We observed performance improvement by the effect of frequency debiasing in both the GloVe and BERT experiments. Notably, the performance of `GloVe-Fdeb` was 7.7 points higher than that of `GloVe` in the test set.

(iii) `GloVe-Fdeb-wh` had no significant improvement over `GloVe-Fdeb`.

(iv) Unlike the GloVe results, the performance of `BERT-Fdeb-wh` was 5.8 points higher than that of `BERT-Fdeb` on the test set.

The observation of result (i) is consistent with the results from a previous study (Huang et al., 2021). Regarding result (ii), the gain from frequency debiasing for GloVe was smaller than that for BERT. We assumed that this was because GloVe had a stronger frequency bias, as suggested by the plot of the PCA coefficients in Section 5.3. To verify this assumption, we present an analysis in Section 5.6.

Regarding results (iii) and (iv), if the effects of whitening and frequency debiasing are independent, then respective gains can be expected when both are applied to the model. However, result (iii) indicates that no significant difference between `GloVe-Fdeb` and

`GloVe-Fdeb-wh` was observed for SWE. This observation reveals that the effect of whitening on SWE is almost the same as that of frequency debiasing, or that there is a large overlap between the two. In contrast, result (iv) indicates that on CWE, whitening has effects other than frequency debiasing, such as the correction of defects inherent in CWE that do not exist in SWE. We speculate that one of these effects may be the correction of the outlier problems reported by Kovaleva et al. (2021); Luo et al. (2021).

5.6 Analysis

In Section 5.5, we assume that whitening has effects on CWE other than frequency debiasing only. In this section, we describe a more in-depth analysis that we conduct to support this assumption. Specifically, we investigate how the quality of embeddings in each model varies with the frequency of words in the sentence. Given a sentence pair p in the STS dataset, we compute the average word frequency ranks defined as follows:

$$R_{\text{sent}}(p) = \frac{1}{N} \sum_{w} r(w), \tag{5.10}$$

where N is the number of words in p, w represents a word in p, and $r(\cdot)$ is a function that takes a word w and returns the frequency rank of w in a predefined vocabulary. Based on R_{sent}, we sort the sentence pairs in the STS evaluation data, and exclude sentence pairs with the highest R_{sent} one by one, and then we evaluate the models on each subset of the dataset. By this procedure, we reduce an average R_{sent} across the dataset, which is defined as follows:

$$R_{\text{dataset}} = \frac{1}{P} \sum_{p} R_{\text{sent}}(p), \tag{5.11}$$

where P is the number of pairs in the dataset. A Higher R_{dataset} signifies that the dataset contains more rare words, while a lower R_{dataset} signifies that the dataset contains more frequent words.

Figure 5.6 presents graphs of the STS performance when varying R_{dataset} for the STS-14 dataset. We observed that for `GloVe`, the larger the $R_{dataset}$, the lower the performance. This indicates that the embeddings of rare words in GloVe had a negative impact on STS performance. In contrast, we did not observe this tendency for BERT. This suggests that the embeddings of rare words in BERT had a small effect on the STS performance. This supports the fact that the representation learning of BERT, which separates infrequent words into subwords, was effective. As mentioned in Section 5.5, we observed significant improvements when whitening was applied to CWE, whereas the impact of frequency debiasing was limited. Thus, we conclude that whitening has other effects other than frequency debiasing.

5.7 Discussion

In this section, we discuss the limitations of our work. In the experiment in Section 5.5, we investigated the relationship between the effects of whitening and frequency debiasing by comparing the performance on the STS task. However, there may be the effects of these methods that are reflected in STS performance and effects that are not. To get a complete picture of the effects of whitening, it is necessary to evaluate them on various tasks.

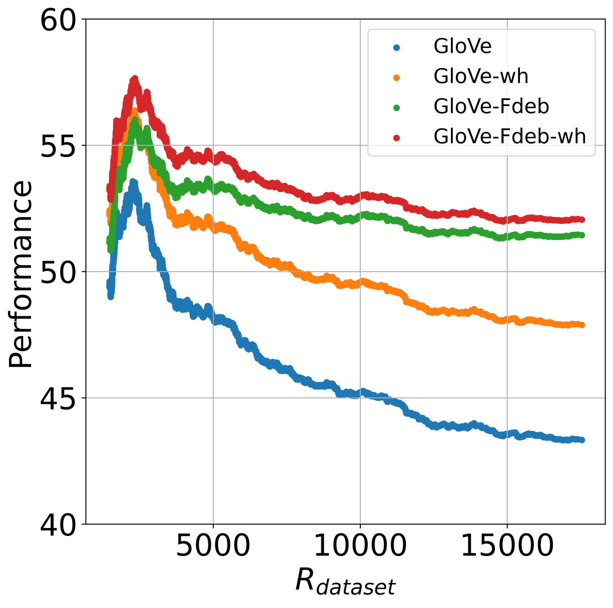
In addition, the effects of the RFD method used in the experiments must be carefully discussed. The conclusion of this chapter assumes that the RFD is working as expected; (i) the frequency debiasing is fully effective and (ii) it has only frequency debiasing influence. Since our proposed RFD is based on existing successful work (Gong et al., 2018), we think these assumptions are not too strong. However, (i) and (ii) assumptions need to be justified through multifaceted observations.

5.8 Conclusion

In this study, we investigated the effect of whitening on SWE and CWE. In a preliminary experiment, we confirmed the existence of a frequency bias in SWE and CWE by visualizing the PCA coefficients of the embeddings and proposed that the application of whitening can remove the bias in the embeddings. In our main experiment, we empirically examined whether whitening had effects other than the effect of frequency debiasing. The results indicated that there was a large overlap between the effects of whitening and the effect of frequency debiasing on SWE. However, on CWE, whitening had effects other than frequency debiasing only.

The main contributions of this work are the discovery of differences in the effects of whitening on GloVe and BERT, and the suggestion that whitening has effects on CWE other than the removal of frequency bias, which is the most studied cause of anisotropy in word embeddings. It remains for future work to identify the remaining effects of whitening and to investigate whether the results of this study can be generalized to other SWE and CWE[3].

[3]Our codes are available at https://github.com/losyer/whitening_effect



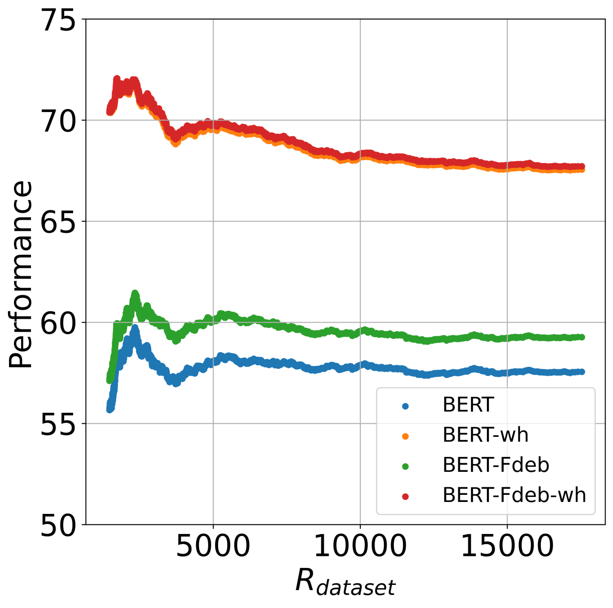


Figure 5.6 Performance/R_{dataset} curves when using models based on GloVe (left) and BERT (right). The x-axis represents R_{dataset}, while the y-axis represents the STS performance evaluated by the Spearman rank correlation.

Chapter 6

Conclusion

In this dissertation, we addressed following research issues:

The data size problem: can low-resource languages attain high-resource-level performance? The learning of text encoders are notoriously data-hungry, meaning that a large amount of data is needed to train the model. A handful of languages may have adequate data, but many more do not have that privilege, limiting the performance of NLP models using text encoders in those languages. In cases where simply increasing the amount of data is not possible, an alternative solution is needed to improve the quality of text encoders.

Memory requirements vs. vocabulary coverage. The strongest text encoders are also usually very large—unrealistically so for the capacity of devices such as mobile phones or IoT appliances which require their use. A naive approach to reducing word embeddings' memory requirement is limiting their vocabulary coverage. However, an unknown word (OOV; out-of-vocabulary) problem then becomes critical, which reduces their usefulness in real-world applications where diverse vocabularies are expected. There is a trade-off between reducing the required amount of memory and the capacity to deal with unknown words. Thus, a balanced approach that reduces the memory requirement while maintaining the ability to handle OOV words would improve the applicability of text encoders.

Bias in embeddings. Various undesired biases in word embeddings can negatively impact a model's task performance. Even leaving the impact on performance aside, biased models can be problematic for real-world applications. Various methods have been proposed to mitigate undesired biases. One of them, *whitening*, has attracted attention due to being simple yet effective in improving performance and mitigating spatial bias. However, while it is clear that whitening mitigates spatial bias in word embeddings, its *semantic* effects (e.g., removal of word frequency bias) are not clear.

The key contributions of this dissertation are summarized as follows:

Investigation of the effectiveness of cross-lingual transfer learning. We proposed a cross-lingual transfer learning method and demonstrated that it improved model performance for low-resource languages. Specifically, as an example of an application of NLP, we conducted experiments on information retrieval tasks. We also clarified the model architecture for information retrieval in which the transfer learning works.

Reducing memory requirements and handling unknown words. In order to simultaneously reduce the memory requirement for static word embeddings and deal with the unknown word problem, we proposed a method that combines memory sharing and key-value-query (KVQ) operation idea. The proposed method succeeded in reducing the amount of required memory while acquiring unknown word embeddings which performed well on word similarity tasks.

Examining the effect of whitening on word embeddings. We examined the semantic effects of *whitening*, one of the isotropization methods for anisotropic embeddings, i.e., reducing spatial bias. From our experiments on a sentence similarity task, we reported that the effect on static word embeddings had significant overlap with the effect of removing word frequency bias, and that this was not the case for contextualized embeddings.

References

(2007). NII test collection for IR systems project. http://research.nii.ac.jp/ntcir/ntcir-ws6/ws-en.html.

(2013). Forum for information retrieval evaluation. https://www.isical.ac.in/~fire/2013/index.html.

(2016). Conference and labs of the evaluation forum. http://clef2016.clef-initiative.eu/.

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. (2009). A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 19–27.

Bahdanau, D., Bosc, T., Jastrzebski, S., Grefenstette, E., Vincent, P., and Bengio, Y. (2017). Learning to Compute Word Embeddings On the Fly. *arXiv preprint arXiv:1706.00286.*

Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., Chapelle, O., and Weinberger, K. (2010). Learning to rank with (a lot of) word features. *Information Retrieval*, 13(3):291–314.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Bruni, E., Tran, N. K., and Baroni, M. (2014). Multimodal Distributional Semantics. *Journal of Artificial Intelligence Research*, 49(1):1–47.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics.

Chen, Q., Zhu, X., Ling, Z.-H., Wei, S., Jiang, H., and Inkpen, D. (2017). Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1657–1668.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223. PMLR.

Dehghani, M., Zamani, H., Severyn, A., Kamps, J., and Croft, W. B. (2017). Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 65–74. ACM.

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

Dong, L. and Lapata, M. (2018). Coarse-to-Fine Decoding for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 731–742.

Ethayarajh, K. (2019). How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65. Association for Computational Linguistics.

Gómez-Rodríguez, C. and Vilares, D. (2018). Constituent Parsing as Sequence Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1314–1324.

Gong, C., He, D., Tan, X., Qin, T., Wang, L., and Liu, T.-Y. (2018). FRAGE: Frequency-agnostic word representation. In *Advances in Neural Information Processing Systems*, volume 31, page 12 pages. Curran Associates, Inc.

Groschwitz, J., Lindemann, M., Fowlie, M., Johnson, M., and Koller, A. (2018). AMR Dependency Parsing with a Typed Semantic Algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1831–1841.

Guo, J., Fan, Y., Ai, Q., and Croft, W. B. (2016). A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64. ACM.

Harris, Z. S. (1954). Distributional structure. <i>WORD</i>, 10(2-3):146–162.

Herbelot, A. and Baroni, M. (2017). High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 304–309.

Hill, F., Reichart, R., and Korhonen, A. (2014). SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *arXiv preprint arXiv:1408.3456*.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 873–882.

Huang, J., Tang, D., Zhong, W., Lu, S., Shou, L., Gong, M., Jiang, D., and Duan, N. (2021). WhiteningBERT: An easy unsupervised sentence embedding approach. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 238–244. Association for Computational Linguistics.

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

Hui, K., Yates, A., Berberich, K., and de Melo, G. (2017). PACRR: A position-aware neural ir model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058. Association for Computational Linguistics.

Kaneko, M. and Bollegala, D. (2022). Unmasking the mask – evaluating social biases in masked language models. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, page 13.

Kiela, D., Bartolo, M., Nie, Y., Kaushik, D., Geiger, A., Wu, Z., Vidgen, B., Prasad, G., Singh, A., Ringshia, P., Ma, Z., Thrush, T., Riedel, S., Waseem, Z., Stenetorp, P., Jia, R., Bansal, M., Potts, C., and Williams, A. (2021). Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Kovaleva, O., Kulshreshtha, S., Rogers, A., and Rumshisky, A. (2021). BERT busters: Outlier dimensions that disrupt transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3392–3405. Association for Computational Linguistics.

Labeau, M. and Allauzen, A. (2017). Character and subword-based word representation for neural language modeling prediction. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 1–13.

Li, B., Zhou, H., He, J., Wang, M., Yang, Y., and Li, L. (2020). On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130. Association for Computational Linguistics.

Liang, Y., Cao, R., Zheng, J., Ren, J., and Gao, L. (2021). Learning to remove: Towards isotropic pre-trained bert embedding. In *Artificial Neural Networks and Machine Learning –ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part V*, pages 448–459. Springer-Verlag.

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

Luo, Z., Kulmizev, A., and Mao, X. (2021). Positional artefacts propagate through masked language model embeddings. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5312–5327. Association for Computational Linguistics.

Luong, T., Socher, R., and Manning, C. (2013). Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the 17th Conference on Computational Natural Language Learning (CoNLL)*, pages 104–113.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3111–3119. Curran Associates Inc.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013c). Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013d). Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 746–751.

Miller, G. A. and Charles, W. G. (1991). Contextual Correlates of Semantic Similarity. *Language & Cognitive Processes*, 6(1):1–28.

Mitra, B., Diaz, F., and Craswell, N. (2017). Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299. International World Wide Web Conferences Steering Committee.

Mu, J. and Viswanath, P. (2018). All-but-the-top: Simple and effective postprocessing for word representations. In *Proceedings of International Conference on Learning Representations*, page 25.

Nicholson, J. (2022). The gender bias inside gpt-3. https://medium.com/madebymckinney/the-gender-bias-inside-gpt-3-748404a3a96c.

Nie, J.-Y. (2010). *Cross-Language Information Retrieval*. Morgan & Claypool Publishers.

Pang, L., Lan, Y., Guo, J., Xu, J., and Cheng, X. (2016). A study of match pyramid models on ad-hoc retrieval. In *Neu-IR '16 SIGIR Workshop on Neural Information Retrieval*.

Pennington, J., Socher, R., and Manning, C. (2014a). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pennington, J., Socher, R., and Manning, C. (2014b). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 2227–2237.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Pilehvar, M. T. and Collier, N. (2017). Inducing Embeddings for Rare and Unseen Words by Leveraging Lexical Resources. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 388–393.

Pinter, Y., Guthrie, R., and Eisenstein, J. (2017). Mimicking Word Embeddings using Subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 102–112.

Prokhorov, V., Pilehvar, M. T., Kartsaklis, D., Liò, P., and Collier, N. (2019). Unseen word representation by aligning heterogeneous lexical semantic spaces. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 6900–6907.

Radinsky, K., Agichtein, E., Gabrilovich, E., and Markovitch, S. (2011). A Word at a Time: Computing Word Relatedness Using Temporal Semantic Analysis. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 337–346.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rajaee, S. and Pilehvar, M. T. (2021). A cluster-based approach for improving isotropy in contextual embedding space. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 575–584. Association for Computational Linguistics.

Ranzato, M., Krizhevsky, A., and Hinton, G. (2010). Factored 3-way restricted boltzmann machines for modeling natural images. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 621–628. PMLR.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics.

Rubenstein, H. and Goodenough, J. B. (1965). Contextual Correlates of Synonymy. *Commun. ACM*, 8(10):627–633.

Sasaki, S., Suzuki, J., and Inui, K. (2019). Subword-based Compact Reconstruction of Word Embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 3498–3508.

Schamoni, S., Hieber, F., Sokolov, A., and Riezler, S. (2014). Learning translational and knowledge-based similarities from relevance rankings for cross-language retrieval. In *Proceedings of the 52 Annual Meeting of the Association for Computational Linguistics*.

Schäuble, P. and Sheridan, P. (1997). Cross-language information retrieval (CLIR) track overview. In *Proceedings of TREC Conference*.

Schick, T. and Schütze, H. (2020). BERTRAM: Improved word embeddings have big impact on contextualized model performance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3996–4007.

Schick, T. and Schütze, H. (2020). Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 8766–8774.

Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014). A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Sokolov, A., Jehl, L., Hieber, F., and Riezler, S. (2013). Boosting cross-language retrieval by learning bilingual phrase associations from relevance rankings. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1688–1699. Association for Computational Linguistics.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Strubell, E., Verga, P., Andor, D., Weiss, D., and McCallum, A. (2018). Linguistically-Informed Self-Attention for Semantic Role Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5027–5038.

Su, J., Cao, J., Liu, W., and Ou, Y. (2021). Whitening sentence representations for better semantics and faster retrieval. *CoRR*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Suzuki, J. and Nagata, M. (2016). Learning Compact Neural Word Embeddings by Parameter Space Sharing. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2046–2052.

Suzuki, J., Takase, S., Kamigaito, H., Morishita, M., and Nagata, M. (2018). An Empirical Study of Building a Strong Baseline for Constituency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 612–618.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning (CoNLL)*, pages 142–147.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics.

Xiong, C., Dai, Z., Callan, J., Liu, Z., and Power, R. (2017). End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64. ACM.

Yu, N., Zhang, M., and Fu, G. (2018). Transition-based Neural RST Parsing with Implicit Syntax Features. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 559–570.

Zhang, B. H., Lemoine, B., and Mitchell, M. (2018). Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340.

Zhao, J., Mudgal, S., and Liang, Y. (2018). Generalizing Word Embeddings using Bag of Subwords. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 601–606.

Zhao, J., Wang, T., Yatskar, M., Cotterell, R., Ordonez, V., and Chang, K.-W. (2019). Gender bias in contextualized word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 629–634. Association for Computational Linguistics.

List of Publications

Journal Papers (Refereed)

1. Shota Sasaki, Jun Suzuki and Kentaro Inui. Subword-Based Compact Reconstruction for Open-Vocabulary Neural Word Embeddings. In IEEE/ACM Transactions on Audio, Speech, and Language Processing, Vol.29, pp.3551-3564, November 2021.

International Conference Papers (Refereed)

1. Kazuaki Hanawa*, Shota Sasaki*, Hiroki Ouchi, Jun Suzuki and Kentaro Inui. The Sally Smedley Hyperpartisan News Detector at SemEval-2019 Task 4. In Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval), pp.1057-1061, June 2019. (* Equal contribution)

2. Shota Sasaki, Jun Suzuki and Kentaro Inui. Subword-based Compact Reconstruction of Word Embeddings. In Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019), pp. 3498-3508, June 2019.

3. Shota Sasaki, Shuo Sun, Shigehiko Schamoni, Kevin Duh and Kentaro Inui. Cross-lingual Learning-to-Rank with Shared Representations. In Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018), pp.458-463, June 2018.

4. Shota Sasaki, Sho Takase, Naoya Inoue, Naoaki Okazaki, and Kentaro Inui. Handling Multiword Expressions in Causality Estimation. In Proceedings of the 12th International Conference on Computational Semantics (IWCS), 6 pages, September 2017.

Non-archival articles (Refereed)

1. Hiroaki Funayama, Shota Sasaki, Yuichiroh Matsubayashi, Tomoya Mizumoto, Jun Suzuki, Masato Mita and Kentaro Inui. Preventing Critical Scoring Errors in Short Answer Scoring with Confidence Estimation. In Proceedings of the 2020 ACL Student Research Workshop (2020 ACL SRW, Non-archival), pp.237–243, July 2020.

Awards

1. 第 13 回 NLP 若手の会シンポジウム奨励賞
2. 言語処理学会第 24 回年次大会若手奨励賞
3. 情報処理学会第 249 回自然言語処理研究会優秀研究賞
4. EMNLP 2021 Outstanding Reviewer

Other publications (Non-refereed)

1. 佐藤俊，大内啓樹，塙一晃，佐々木翔大，乾健太郎. 事例ベース推論を行うニューラルモデルの説明性とハブ現象の関係. 情報処理学会第 249 回自然言語処理研究会 (NL 研)，7 月 2021.

2. 佐藤俊，大内啓樹，佐々木翔大，塙一晃，乾健太郎. 説明性の高いニューラルモデルの予測確信度に関する分析. 言語処理学会第 27 回年次大会，pp.1204-1209，3 月 2021.

3. 佐々木翔大，大内啓樹，鈴木潤，Ana Brassard，乾健太郎．単一評価サンプルのためのトランズダクティブ学習．言語処理学会第 26 回年次大会，3 月 2020.

4. 舟山弘晃，佐々木翔大，水本智也，三田雅人，鈴木潤，松林優一郎，乾健太郎．記述式答案自動採点のための確信度推定手法の検討．言語処理学会第 26 回年次大会，3 月 2020.

5. 佐藤俊，大内啓樹，塙一晃，佐々木翔大，乾健太郎. 訓練過程における予測ラベルの遷移頻度情報を用いた予測確信度計算手法の改善. 第 15 回 NLP 若手の会シンポジウム (YANS)，9 月 2020.

6. 佐藤俊，佐々木翔大，大内啓樹，鈴木潤，乾健太郎．評価データのクラスタリングを用いた記述式答案自動採点のためのトランズダクティブ学習. 言語処理学会第 26 回年次大会，3 月 2020.

7. 舟山弘晃，佐々木翔大，水本智也，三田雅人，鈴木潤，乾健太郎．自動採点における確信度推定手法．第 14 回 NLP 若手の会シンポジウム，8 月 2019.

8. 中村拓，田然，佐々木翔大，乾健太郎．単語埋め込みにおける複数視点の対義語判定．2019 年度人工知能学会全国大会 (第 33 回)，4 pages，6 月 2019.

9. 佐々木翔大，鈴木潤，乾健太郎．サブワードに基づく単語分散表現の縮約モデリング．言語処理学会第 25 回年次大会，3 月 2019.

10. 佐々木翔大，鈴木潤，乾健太郎．サブワードに基づく単語ベクトルの再構築．第 13 回 NLP 若手の会シンポジウム，8 月 2018. Shota Sasaki, Shuo Sun, Shigehiko Schamoni, Kevin Duh and Kentaro Inui. Cross-lingual Information Retrieval with Shared Representations. The 5th CWRU-TOHOKU Joint Workshop, 8 月 2018.

11. 佐々木翔大，Shuo Sun，Shigehiko Schamoni，Kevin Duh，乾健太郎．言語横断的情報検索の大規模データセットとパラメータ共有モデル．言語処理学会第24回年次大会，3月2018.

12. 佐々木翔大，田然，乾健太郎．数量表現と比較に着目した意味解析に向けて．第12回NLP若手の会シンポジウム，9月2017.

13. 佐々木翔大，高瀬翔，井之上直也，岡崎直観，乾健太郎．複単語表現を利用した因果関係推定モデルの改善．第231回自然言語処理研究会・第116回音声言語情報処理研究会，5月2017.