Doctoral Dissertation

Semantic Operations on an Embedding Space

Yoichi Ishibashi

Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Satoshi Nakamura
Augmented Human Communication Lab.
(Division of Information Science)

Submitted on March 17, 2023

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of Engineering

Yoichi Ishibashi

Thesis Committee:

Professor Satoshi Nakamura (Supervisor)
Professor Taro Watanabe (Co-supervisor)
Professor Danushka Bollegala (The University of Liverpool)
Associate Professor Katsuhito Sudoh (Co-supervisor)

Semantic Operations on an Embedding Space*

Yoichi Ishibashi

Abstract

This thesis addresses the formulation of semantic representation and operations in an embedding space. Word embeddings can mathematically represent ambiguous and diverse information in language. In a pre-trained embedding space, it is known that the semantic operation such as $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} = \overrightarrow{queen}$, also called additive compositionality, can be computed. Semantic operations have various practical advantages such as generality and are worth studying for our understanding of the nature of the embedding space. However, no semantic operations other than additive compositionality have been discovered. In this thesis, we formulated two types of semantic operations to tackle this limitation.

First, we propose a *binary attribute transfer* that inverts the binary attributes of words. An analogy of word vectors can transfer word attributes, but it requires explicit knowledge of whether the input word has the attribute or not (e.g., the gender attribute of "king" is male). However, this knowledge cannot be developed for various words and attributes in practice. We define an ideal transfer function without using the knowledge and propose *reflection*-based word attribute transfer. We demonstrate that this method achieves high accuracy in transferring words with the binary attribute to be transferred and high stability in not transferring other words.

Next, we formulate *word sets* and *set operations* in the pre-trained embedding space. Set operations are critical because they can be general-purpose tools in natural language processing. We propose subspace-based formulations inspired by *quantum logic*. We quantitatively and qualitatively demonstrate that the proposed method is valid as a semantic operation using a word set dataset and

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, March 17.

that our proposed set operation improves performance on downstream tasks such as sentence similarity.

Keywords:

natural language processing, representation learning, embedding, reflection, subspace, pre-trained model

Contents

List of Figures

List of Tables

1 Introduction

1.1 Representation of Meaning in Language

The meaning of natural language is complex and ambiguous information. We humans realize sophisticated information exchange by understanding this information. For example, we can communicate with each other by speaking and writing language. The meaning of language can be represented by symbols, such as letters, words, and sentences. We perform sophisticated information processing by representing the meaning of language in such a way. Natural language processing (NLP) aims to process such natural language information processing by computer.

Recent breakthroughs in deep learning have led to tremendous progress in NLP, computer vision, and speech processing [18, 23, 98]. It has achieved phenomenal performance on a variety of NLP tasks. For example, a recent large pre-trained language model can generate very natural, human-like sentences [18]. More recently, approaches have been studied that use a single model to learn many tasks. Such a model, called a foundation model [15], can be used generically for a variety of tasks, and has been reported to exceed average human performance on more than 150 tasks [23]. The key to these great successes in NLP lies in an approach that automatically learns features of data. Feature representation of language meaning has become an integral part of modern NLP.

However, it is not easy to represent the meaning. For example, the following two sentences are not at all similar on the surface of the sentence because they have no words in common, but their meanings are very similar.

$$\begin{array}{c} \textit{Those dogs are cute.} \\ \textit{That puppy is adorable!} \end{array} \tag{1.1}$$

The following two sentences are superficially similar because they share many common words, but their meanings differ.

$$\begin{aligned} &\textit{He is a boy.} \\ &\textit{She is not a boy.} \end{aligned} \tag{1.2}$$

As in these examples, it is not easy for a machine to determine the similarity of meaning from only the surface information of a sentence. On the other hand, we humans can easily recognize these differences in meaning. In NLP, such human intuition needs to be represented mathematically. In other words, the goal of representation learning in a language is to create a computational model that reflects such human intuition.

1.2 Symbolic Representation

Semantic representation methodologies can be classified into two categories: (1) symbolic representation and (2) vector representation. Let us review here the history of semantic representation to highlight the importance of vector representation. NLP has been studied for more than 70 years. It began in 1949 with Weaver's Memorandum [106]. This memorandum, which is entitled simply *Translation*, formulated the methodology and goals of machine translation. Later, a project known as *The Georgetown Experiment* successfully translated over 60 sentences from Russian into English [46]. These studies triggered a worldwide interest in machine translation research. However, the performance was limited. Because their machine translation system was based on a dictionary and word order. Such a simple method could not adequately represent the ambiguous and diverse information in natural language. Researchers have been working on this problem for a long time. The representation of meaning remains the most important research problem in NLP.

In 1957, Chomsky's publication of generative grammar [22] led to the incorporation of linguistics into machine translation. Since then, many researchers began to study methods of representing meaning. By the early 1970s, many semantic representations were proposed, including case grammar [32] and semantic networks [26].

In the 1980s, the symbolic approach became mainstream and achieved remarkable results. For example, methods for representing or reasoning knowledge using symbolic logic were proposed [105]. Symbolic approaches were employed in expert systems [105], tokenization, parsing, etc., and solved many problems in NLP. At that time, semantic representation was used in a straightforward way, dealing with tokens, words, phrases, and sentences as they are. Such symbolic representations have the advantage of being easily implemented as well as easily understood. On the other hand, symbolic representation does not consider the ambiguity of meaning inherent in natural language.

1.3 Vector Representation

In this context, *distributed representation*, in which information is not held locally but distributed, was proposed by Hinton et al [44]. Distributed representations can mathematically represent ambiguous and diverse information in language. While symbolic representations are difficult to represent or cannot cover many cases, distributed representations have solved them. With recent improvements in computational performance, there has been an increase in research into distributed representation. Bengio et al. [11] proposed a neural probabilistic language model for learning word distributed representations. Mikolov et al. [74] made it possible to learn from a large corpus of texts by improving computational efficiency.

Currently, *vector representation* is the mainstream distributed representation because it is compatible with deep learning algorithms. In NLP, vector representations are used to represent words [74], sentences [90], etc.

Vector representation has various advantages. By representing linguistic features as vectors, we can easily compute the similarity of words and sentences [90]. One of the other advantages of using vector representations is that we can define a mapping between vectors. For example, a translation from German to English can be defined as a mapping from German tokens to English tokens [98]. The approach of using neural networks to learn this mapping has led to significant developments in NLP. It has been applied with great success to various tasks such as machine translation [98] dialogue systems [102], text classification [30],

and semantic textual similarity [4].

1.4 Semantic Operations in Vector Representation

In a pre-trained word vector space, it is known that the following *semantic operation* can be performed [63].

$$\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} = \overrightarrow{queen} \tag{1.3}$$

Interestingly, this operation is very close to human intuition, i.e., the analogy can be computed that results in "*man* is to *woman* as *king* is to *queen*", by adding and subtracting vectors. Such a property is called *additive compositionality*. This kind of semantic operation has two major unique properties.

- First, the semantic operation is highly versatile. The most typical application of additive composition is the computation of sentence vectors; in NLP, it is a frequent practice to compute sentence vectors by averaging word vectors [75, 90]. The additive construction is a simple but good way to obtain representative vectors. For example, attention mechanism [10,98], which is a weighted average of a set of feature vectors, has become a major technique in recent language models.

  The semantic operation has been widely used in various applications. Bordes et al. [16] proposed the loss function based on additive compositionality for learning entity and relation embeddings of a knowledge graph. Reed et al. [89] use an analogy for changing the attributes of an image.

  Besides, for semantic operations that work in a pre-trained embedding space, semantic operations can be used without additional task-specific training. This is because they are operations that are valid in a pre-trained embedding space. Based on this property, we can use an analogy to evaluate an embedding space [63, 76]. Mikolov et al. [76] proposed an analogy-based evaluation benchmark for word embeddings. This is based on the idea that the more analogies an embedding space consist of, the better the embedding

space is at reproducing human intuition, including semantic and syntactic analogies, as shown in Figure 1.1.

- Finally, semantic operations can help us understand embedding spaces. Operations such as Equation 1.3 can be intuitively interpreted as the addition and subtraction of concepts. Recent theoretical studies reveal that additive compositionality is closely related to probability theory and information theory [7, 31, 38, 64].

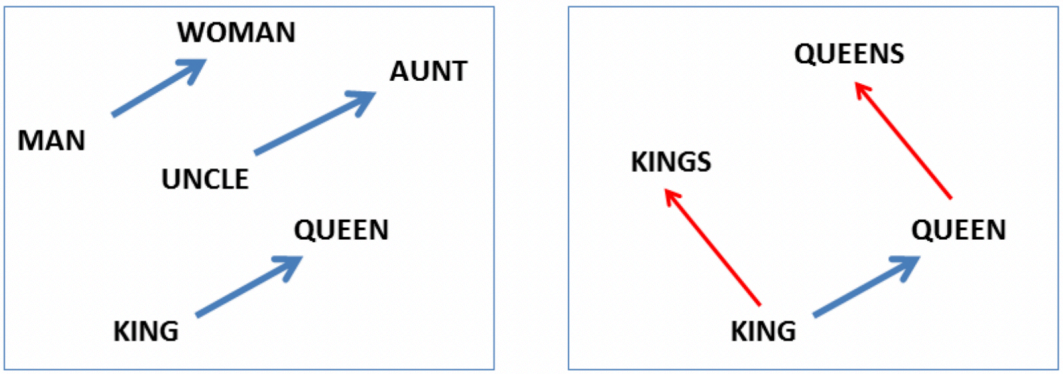


Figure 1.1: Example of analogy in word embedding space. The figure is taken from [76]

1.5 Scope of This Thesis

As mentioned above, semantic operation has various practical advantages and is worth studying for our understanding of the nature of embedding spaces. However, no semantic operations other than additive compositionality have been studied.

In this thesis, we formulate novel semantic operations. We formulate two new types of semantic operations, (1) *word attribute transfer*:

$$\begin{aligned} \overrightarrow{king} &= f_{gender}(\overrightarrow{queen}), \\ \overrightarrow{queen} &= f_{gender}(\overrightarrow{king}), \\ \overrightarrow{apple} &= f_{gender}(\overrightarrow{apple}), \end{aligned} \tag{1.4}$$

and (2) *set operations* such as intersection ($\cap$), and set membership ($\in$):

$$\begin{aligned} \text{Color} &= \{\overrightarrow{red}, \overrightarrow{blue}, \overrightarrow{green}, \overrightarrow{orange}, \dots\}, \\ \text{Fruit} &= \{\overrightarrow{apple}, \overrightarrow{orange}, \overrightarrow{peach}, \dots\}, \\ \overrightarrow{orange} &\in \text{Color} \cap \text{Fruit}, \end{aligned} \tag{1.5}$$

and set similarity:

$$\begin{aligned} \text{A} &= \{\overrightarrow{Those}, \overrightarrow{dogs}, \overrightarrow{are}, \overrightarrow{cute}\}, \\ \text{B} &= \{\overrightarrow{That}, \overrightarrow{puppy}, \overrightarrow{is}, \overrightarrow{adorable}\}, \\ \text{SentenceSimilarity} &= \text{SetSimilarity}(\text{A}, \text{B}), \end{aligned} \tag{1.6}$$

and demonstrate that our semantic operations work well in embedding space and have high performance in applications.

1.6 Outline

The rest of this thesis is organized as follows:

Embedding (Chapter 2) We review current word embedding methodologies in two major categories.

Word Attribute Transfer (Chapter 3) We propose a semantic operation called *reflection-based word attribute transfer*. Word attribute transfer is a mapping that inverts attributes such as gender and can be used to expand sentence data. For example, we obtain a new sentence *She is a girl.* by transferring the gender attribute of each word in *He is a boy.* We formulate word attribute transfer on the embedding space. A straightforward method of word attribute transfer is to use analogies by word vectors such as Equation 1.3. For example, to invert gender, $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman}$ can be converted to $\overrightarrow{queen}$. However, the analogy-based method assumes that the gender attribute of the input word is known in advance since whether the vector is added or subtracted depends on whether the input word is male or female. Such knowledge is innumerable and cannot be manually assigned to every word. In this chapter, (1) we formulate a function that does not use knowledge of attributes, and (2) we propose a word attribute transfer based on *reflection*, which is one such ideal mapping. Experimental results show that our reflection-based method is more stable than other semantic operations for word attribute transfer.

Subspace-based Set Operations (Chapter 4) Word embedding is often exploited for tasks using sets of words, although standard methods for representing word sets and set operations remain limited. If we can leverage the advantage of word embedding for such set operations, we can calculate sentence similarity and find words that effectively share a concept with a given word set straightforwardly. In this study, we formulate representations of sets and set operations in a pre-trained word embedding space. Inspired by *quantum logic*, we propose a novel formulation of set operations using subspaces in a pre-trained word embedding space. Based on our definitions, we propose two metrics based on the degree to which a word belongs to a set and the similarity between embedding two sets. Our experiments with Text Concept Set Retrieval and Semantic Textual Similarity tasks demonstrated the effectiveness of our proposed method.

Conclusion and Future Directions (Chapter 5) This chapter summarizes the contribution of the thesis work and discusses future directions.

2 Word Embedding

2.1 Vector Representation

2.1.1 One-hot Representation

One-hot representation is the most intuitive and easiest way to represent a word as a vector. It represents a word by a one-hot vector in which only one element has a value of one and the other elements are zero. In the case of word representation, the number of dimensions of a one-hot vector is equal to the number of words in a vocabulary in order to make each element correspond to one word. For example, each word in the vocabulary V = $\{king, man, queen, woman\}$ can be represented as a one-hot vector as follows:

$$\overrightarrow{king} = (1, 0, 0, 0), \tag{2.1}$$

$$\overrightarrow{man} = (0, 1, 0, 0), \tag{2.2}$$

$$\overrightarrow{queen} = (0, 0, 1, 0), \tag{2.3}$$

$$\overrightarrow{woman} = (0, 0, 0, 1). \tag{2.4}$$

There are two main problems with one-hot representation. One is the inability to represent semantic similarity. This is due to the fact that one-hot vectors are sparse (discrete) representations. All one-hot vectors are orthogonal, so the similarity of each word is not represented. Another problem is memory inefficiency. When the vocabulary is large, the dimensions of the one-hot vectors also increase. This results in processing a huge matrix. For example, if the size of the vocabulary is 100000, the matrix has 10 billion elements. We need huge memory usage and computation time.

2.1.2 Distributed Representation

There have been many studies on low-dimensional vector representations that capture the similarity between words to solve this. Distributed representation [44] is one of these methods. The word vectors based on the distributed representation have been successfully reduced to several hundred dimensions, independent of the vocabulary size. In the next section, we discuss word embedding, the most popular method of word distributed representation, and in sections section 2.3 and section 2.4, we describe several word embeddings in detail.

2.2 Word Embedding and Semantic Operation

Word embedding is a method of computing a distributed representation of a word. In NLP, embedding is mapping from linguistic features to a vector space. Word embedding maps a word to a vector space, and sentence embedding maps a sentence to a vector space. If similar words are located closely in the vector space, such an embedding can capture the word similarity.

The way to obtain good word embeddings is to learn a word vector from the context based on the distributional hypothesis. The Distributional Hypothesis is that words having similar meanings appear in similar contexts. In other words, the meaning of a word can be determined from its surrounding words. The Distributional Hypothesis is very simple, but word embedding based on the hypothesis can represent the meaning of a word quite well.

Typical word embedding methods include word2vec* [75], Global Vectors (GloVe) [84], and fastText [14]. They represent words as vectors in a Euclidean space. This has the advantage that the degree of similarity between words can be intuitively defined as the angle between their word vectors as follows:

$$\begin{aligned} \text{cos\_similarity}(\boldsymbol{v}_a, \boldsymbol{v}_b) &= \cos\theta, \\ &= \frac{|\boldsymbol{v}_a \cdot \boldsymbol{v}_b|}{\|\boldsymbol{v}_a\|\|\boldsymbol{v}_b\|}, \end{aligned} \tag{2.5}$$

*This term is often used as a generic term for two algorithms (skip-gram and continuous bag-of-words).

where a and b are words, and $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$ are their word vectors, and θ is the angle between $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$.

The word embeddings are not only used to calculate the similarity between words, but they also enable semantic operations such as $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} \approx \overrightarrow{queen}$ (Figure 2.1). Besides such analogy, additive construction also allows for the operation of the degree of the meaning of a word. For example, Kim et al. [56] reported a word vector that is most similar to the center between $\overrightarrow{good}$ and $\overrightarrow{best}$ is $\overrightarrow{better}$ (Figure 2.2). Moreover, a recent study has revealed that additive composition corresponds to word sense operations, i.e., AND (e.g. $king = man \wedge royal$), OR (e.g. $case = box \vee instance$), and NOT (e.g. $hate = \neg love$).

For the vector space, Euclidean space is often used for practical convenience [14, 75, 84]. Sometimes a vector space other than Euclidean space is used to take into account the nature of the language. For example, Nickel et al. [80] and subsequent work [34, 95] employ a hyperbolic space to embed a hierarchical structure of linguistic concepts.

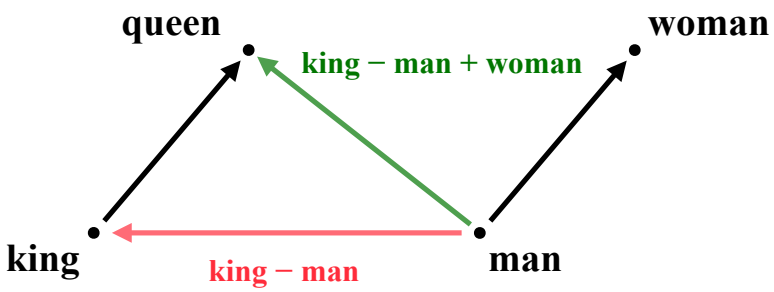


Figure 2.1: PMI-based word embeddings capture analogic relations such as $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} \approx \overrightarrow{queen}$.

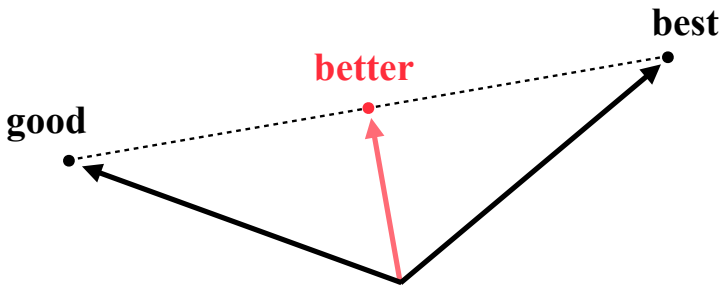


Figure 2.2: $\overrightarrow{better}$ is embedded in the center between $\overrightarrow{good}$ and $\overrightarrow{best}$

2.3 Static Embeddings

In this section, we describe static embedding, which gives just one representation for a word.

2.3.1 word2vec

Word2vec is the most popular word embedding that successfully learns word vectors from large text corpus by improving computational efficiency. To obtain word embeddings, two-layer neural networks are trained on a large corpus to predict the linguistic contexts of words.

Word2vec includes two models: continuous bag-of-words (CBOW) or skip-gram (SG). These models are based on the Distributional Hypothesis [33, 43]. In other words, these learning algorithms are designed to learn a word vector from the context of the word. The goal of CBOW is to predict the center word given the surrounding words. In contrast, SG predicts the surrounding words from the center word.

Let $\mathrm{S} = \{w_1, ..., w_t, ..., w_T\}$ be a sentence, w_t $(1 \leq t \leq T)$ be a word, and let $\mathrm{C} = \{w_{t-\delta}, ..., w_{t-1}, w_{t+1}, ..., w_{t+\delta}\}$ be context words within a fixed-size window determined by δ. SG predicts context words C from a target word w_t. On the other hand, CBOW predicts the target word w_t from context words C. SG maximizes the following log-likelihood of the conditional probability distribution function:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-\delta\leq j\leq\delta, j\neq 0}\log p(w_{t+j} \mid w_t), \tag{2.6}$$

$$p(w_{t+j} \mid w_t) = \frac{\exp(\boldsymbol{w}_{t+j}^{\top}\boldsymbol{w}_t)}{\sum_{v\in \mathrm{V}}\exp(\boldsymbol{v}^{\top}\boldsymbol{w}_t)}, \tag{2.7}$$

where δ is a window size, and $\boldsymbol{v}$ is a word vector of the word v that is a element of a vocabulary set V. Here, we designate a vector of v as a bold letter $\boldsymbol{v}$.

As shown in Eq. 2.7, SG requires repeated calculations for the vocabulary size $|\mathrm{V}|$ for each step of the prediction. Since $|\mathrm{V}|$ is generally 10^5 to 10^7, SG raises the computational cost problem. We can reduce the computational cost by negative

sampling to avoid it. Skip-gram with negative sampling (SGNS) works as well as Skip-gram while reducing the computational cost by classifying the co-occurring pair (w, w') into one and non-co-occurring pair $(w, \tilde{w})$ into zero. In SGNS, the objective is defined as follows:

$$-\log \sigma(\boldsymbol{w'}^{\top} \boldsymbol{w}_t) - \sum_{\tilde{w} \in \mathrm{N}} \log \sigma(-\tilde{\boldsymbol{w}}^{\top} \boldsymbol{w}_t) \tag{2.8}$$

where N is a set of negative samples, and σ is a sigmoid function. Negative samples are randomly sampled from V considering the frequency.

2.3.2 GloVe

Other word embedding methods include Global Vectors (GloVe) [84]. GloVe combines the advantages of the matrix factorization method and the shallow window-based method. GloVe differs from word2vec in that it uses global information. Word2vec uses local co-occurrence information, i.e., the context of words in a sentence. In contrast, GloVe uses information on co-occurrences between words in the entire corpus. GloVe can perform analogy operations as well as word2vec.

The objective is defined as follows:

$$\sum_{i,j=1}^{|\mathrm{V}|} f(X_{i,j})(\boldsymbol{w}_i^{\top} \boldsymbol{w}_j + b_i + b_j - \log X_{i,j})^2, \tag{2.9}$$

where $X_{i,j}$ is the number of co-occurrence of the words w_i and w_j, and b_i and b_j are the bias terms for them. The weighting function f is defined as follows:

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & \text{if } \ x < x_{max}, \\ 1 & \text{otherwise}, \end{cases} \tag{2.10}$$

where the cut-off parameter x_{max} and α are hyperparameters.

2.4 Dynamic Embeddings

In this section, we describe dynamic embedding (contextualized embedding), which gives contextual representation for a word.

Words are often polysemous. Let us consider the following example to clarify the importance of representing the polysemy of a word. The following two sentences share the word *bank* but it has different meanings.

$$\begin{aligned} &\textit{I made a deposit of \$500 at the bank.} \\ &\textit{The river flowed over its bank.} \end{aligned} \tag{2.11}$$

In the first sentence, *bank* means a financial institution, while it means the slope adjacent to a river in the second sentence. The word surfaces are the same, but their semantic representations can be different. Polysemy is one of the key issues in NLP. However, static embedding does not adequately represent this. For example, in word2vec or GloVe, a word vector is unique for a given word. That is, the word vector does not change depending on its context.

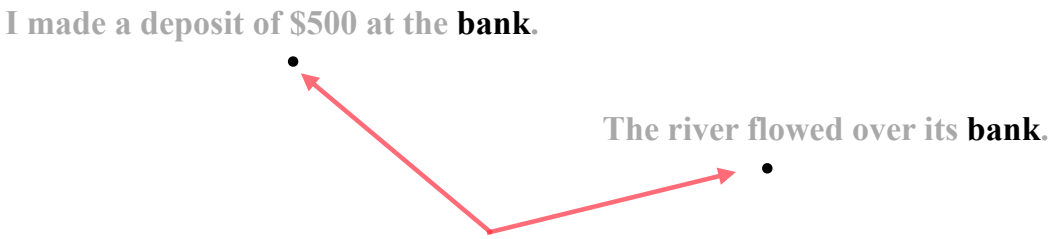


Figure 2.3: Example of dynamic embeddings

Dynamic embedding (also called contextualized embedding) is an embedding in which the word or subword vector changes dynamically depending on the context. ELMo (Embeddings from Language Models) [85] and BERT (Bidirectional Encoder Representations from Transformers) [30] are typical methods that can obtain dynamic embeddings. ELMo consists of bi-directional two-layer LSTMs (Long Short-Term Memory) [45], which look at the whole sentence and give contextualized word vectors. ELMo is trained as a language model. Training is done through self-supervised learning. It does not require labeled training data and can be trained on large amounts of unlabeled text data.

2.4.1 BERT

BERT [30] is a language representation model for NLP. Similar to ELMo, BERT can embed an input text into a contextual vector.

It differs from ELMo in that BERT's model architecture is a Transformer encoder [98]. The self-attention mechanism allows it to obtain superior representations and improve computational efficiency through parallel computation.

The advantage of BERT is that it can achieve high performance on a variety of tasks[†] through general-purpose representations. This is achieved through two training phases as shown in Figure 2.4: (1) pre-training on general domain large-scale training data and (2) tuning pre-trained BERT according to the task (called fine-tuning). In (2), by using the pre-trained BERT as initial weights, it can achieve high performance with a small amount of training data. We often have difficulty preparing large training data sets for specific tasks, but BERT allows us to solve this issue. Moreover, fine-tuning takes only a few epochs. It is no longer necessary to train the model from scratch for a specific task. Such training schemes have been employed in many studies and have become common [70, 88].

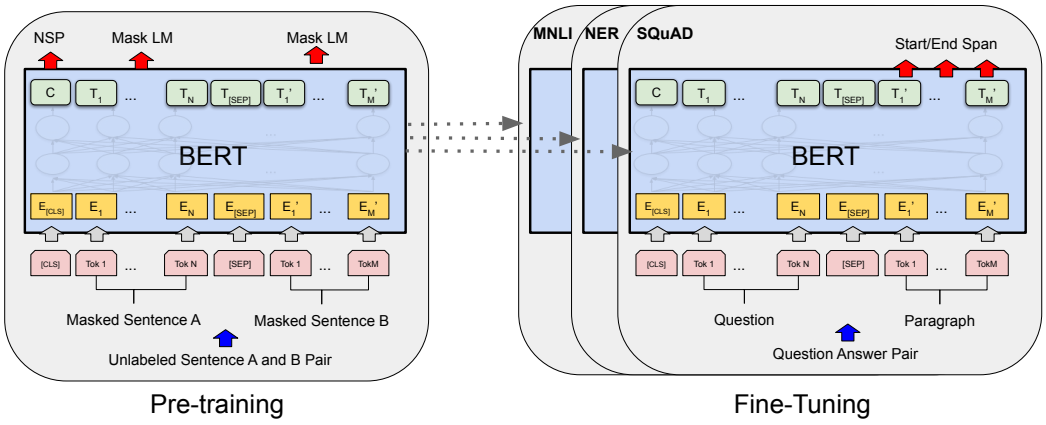


Figure 2.4: Procedure of pre-training and fine-tuning for BERT. The figure is taken from [30]

[†]In machine learning, a *task* is a type of prediction designed to solve a given problem. It can be broadly classified into classification, regression, etc.

3 Word Attribute Transfer

3.1 Background

The distributed representation is compatible with neural networks because the representation can capture the features of language data and compress them into low-dimensional vectors. Word embedding methods handle word semantics in NLP [74,75,80,84,101]. Word embedding models such as SGNS [75] or GloVe [84] capture analogic relations such as $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} \approx \overrightarrow{queen}$. Previous works [6, 7, 31, 38, 64] offer theoretical explanations based on pointwise mutual information (PMI) [25] for maintaining analogic relations in word vectors.

These relations can be used to transfer a certain attribute of a word, such as changing *king* into *queen* by transferring its gender. This transfer can be applied to perform data augmentation; for example, rewriting *He is a boy* to *She is a girl*. It can be used to generate negative examples for natural language inference [55]. For example, in the Natural Language Inference (NLI) [17, 108] corpus, negative examples can be generated by transferring a hypothesis sentence from entailment to contradiction. We tackled a novel task that changes a word by transferring certain attributes associated with the word, which is called *word attribute transfer*.

A naive way for word attribute transfer is to use a difference vector based on analogic relations, such as adding $\overrightarrow{woman} - \overrightarrow{man}$ to $\overrightarrow{king}$ to obtain $\overrightarrow{queen}$. This requires explicit knowledge whether an input word is male or female. We have to add a difference vector to a male word and subtract it from a female word for achieving gender transfer. We also have to avoid changing words that are invariant with respect to gender attributes, such as *is* and *a* in the example above, as they are gender-invariant words. Developing such knowledge is significantly costly for words and attributes in practice. In this thesis, we propose a novel framework for word attribute transfer based on *reflection* that does not require

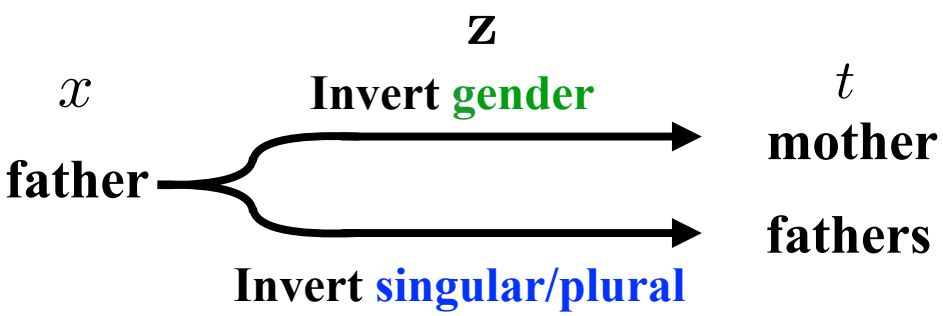


Figure 3.1: Examples of word attribute transfer

explicit knowledge of the given words in its prediction.

The contributions of this work are twofold: (1) We propose a word attribute transfer method that obtains a vector with an inverted binary attribute without explicit knowledge. (2) The proposed method demonstrates more accurate word attribute transfer for words that have target attributes than other baseline methods, while ensuring that the words that do not have target attributes are unchanged.

3.2 Word Attribute Transfer Task

In this task, we focus on modeling the binary attributes (e.g., male and female*). Let x denote a word and let $\boldsymbol{v}_x \in \mathbb{R}^n$ denote its n-dimensional vector representation. We assume that $\boldsymbol{v}_x$ is learned in advance using an embedding model, such as a skip-gram. In this task, we have two inputs, a word x and vector $\boldsymbol{z} \in \mathbb{R}^n$, which represents a certain target attribute, and an output word y. y is the word obtained through the transfer of x according to the target attribute specified by $\boldsymbol{z}$. y should be the same as the reference word t. Note that t is the same as x when x is invariant based on the target attribute. In this thesis, $\boldsymbol{z}$ is an n-dimensional vector embedded from a target attribute ID by using an embedding function of a deep learning framework. For example, given a set of attributes $\mathrm{Z} = \{\mathrm{gender}, \mathrm{antonym}\}$, we assign different random vectors $\boldsymbol{z}_{\mathrm{gender}}$ for gender and $\boldsymbol{z}_{\mathrm{antonym}}$ for antonym. Let A denote a set of triplets $(x, t, \boldsymbol{z})$, e.g.,

*Gender-specific words are sometimes considered socially problematic. Here, we use this as an example based on the man-woman relation.

$(man, woman, \boldsymbol{z}_{\text{gender}}) \in \text{A}_{\text{gender}}$, and N denote a set of invariant words for an attribute $\boldsymbol{z}$, e.g., $(person, \boldsymbol{z}_{\text{gender}}) \in \text{N}_{\text{gender}}$. This task transfers an input word vector $\boldsymbol{v}_x$ to an output word vector $\boldsymbol{v}_y \in \mathbb{R}^n$ by using a transfer function $f_{\boldsymbol{z}_{\text{attr}}}$ that inverts the attribute $\boldsymbol{z}_{\text{attr}}$ of $\boldsymbol{v}_x$. $\boldsymbol{v}_y$ is expected to be the same as its reference word vector $\boldsymbol{v}_t \in \mathbb{R}^n$. This is denoted according to the following formula:

$$\boldsymbol{v}_t \approx \boldsymbol{v}_y = f_{\boldsymbol{z}}(\boldsymbol{v}_x). \tag{3.1}$$

The following properties must be satisfied: (1) attribute words $\{x \mid (x, t, \boldsymbol{z}) \in \text{A}\}$ are transferred to their counterparts, and (2) invariant words $\{x \mid (x, \boldsymbol{z}) \in \text{N}\}$ are not changed (are transferred back into themselves). For instance, with $\boldsymbol{z}_{\text{gender}}$, for a given input word *man*, the gender attribute transfer $f_{\boldsymbol{z}_{\text{gender}}}(\boldsymbol{v}_{man})$ should result in a vector close to $\boldsymbol{v}_{woman}$. When given another input word *person* as x, the result should be $\boldsymbol{v}_{person}$.

3.3 Analogy-based Word Attribute Transfer

Analogy is a general idea that can be used for word attribute transfer. PMI-based word embedding methods, such as SGNS and GloVe, capture analogic relations, as shown in Eq. 3.2 [63, 68, 76]. By rearranging Eq. 3.2, Eq. 3.3 is obtained:

$$\boldsymbol{v}_{queen} \approx \boldsymbol{v}_{king} - \boldsymbol{v}_{man} + \boldsymbol{v}_{woman}, \tag{3.2}$$

$$\approx \boldsymbol{v}_{king} - (\boldsymbol{v}_{man} - \boldsymbol{v}_{woman}). \tag{3.3}$$

The analogy-based transfer function is

$$f_{\boldsymbol{z}}(\boldsymbol{v}_x) = \begin{cases} \boldsymbol{v}_x - \boldsymbol{d} & \text{if } \ x \in \text{M}, \\ \boldsymbol{v}_x + \boldsymbol{d} & \text{if } \ x \in \text{F}, \end{cases} \tag{3.4}$$

where M is a set of words with a particular target attribute (e.g., male) and F is a set of words with an inverse attribute (e.g., female). $\boldsymbol{d}$ is a difference vector, such as $\boldsymbol{v}_{man} - \boldsymbol{v}_{woman}$. Eq. 3.4 indicates that the operation changes depending on whether the input word x belongs to M or F. However, to transfer the word attribute based on analogy, we require explicit knowledge such as the attribute value (M, F, or others) that is contained by the input word.

3.4 Reflection-based Word Attribute Transfer

3.4.1 Ideal Transfer Mapping without Knowledge

What is an ideal transfer function $f_{\boldsymbol{z}}$ for the word attribute transfer? The following are the ideal natures of a transfer function:

$$\forall(m, w, \boldsymbol{z}) \in \mathrm{A}, \quad \boldsymbol{v}_m = f_{\boldsymbol{z}}(\boldsymbol{v}_w), \tag{3.5}$$

$$\forall(m, w, \boldsymbol{z}) \in \mathrm{A}, \quad \boldsymbol{v}_w = f_{\boldsymbol{z}}(\boldsymbol{v}_m), \tag{3.6}$$

$$\forall(u, \boldsymbol{z}) \in \mathrm{N}, \quad \boldsymbol{v}_u = f_{\boldsymbol{z}}(\boldsymbol{v}_u). \tag{3.7}$$

The function $f_{\boldsymbol{z}}$ enables a word to be transferred without explicit knowledge because the operation of $f_{\boldsymbol{z}}$ does not change depending on whether the input word belongs to M or F. By combining Eqs. 3.5, 3.6 and 3.7, we obtain the following formulas:

$$\forall(m, w, \boldsymbol{z}) \in \mathrm{A}, \quad \boldsymbol{v}_m = f_{\boldsymbol{z}}(f_{\boldsymbol{z}}(\boldsymbol{v}_m)), \tag{3.8}$$

$$\forall(m, w, \boldsymbol{z}) \in \mathrm{A}, \quad \boldsymbol{v}_w = f_{\boldsymbol{z}}(f_{\boldsymbol{z}}(\boldsymbol{v}_w)), \tag{3.9}$$

$$\forall(u, \boldsymbol{z}) \in \mathrm{N}, \quad \boldsymbol{v}_u = f_{\boldsymbol{z}}(f_{\boldsymbol{z}}(\boldsymbol{v}_u)). \tag{3.10}$$

Hence, the ideal transfer function is a mapping that becomes an identity mapping when we apply it twice for any $\boldsymbol{v}$. Such a mapping is called *involution* in geometry. For example, $f\colon \boldsymbol{v} \mapsto -\boldsymbol{v}$ is an example of an involution.

3.4.2 Reflection

Reflection $\mathrm{Ref}_{\boldsymbol{a},\boldsymbol{c}}$ is an ideal function because this mapping is an involution, as shown below:

$$\forall \boldsymbol{v} \in \mathbb{R}^n, \quad \boldsymbol{v} = \mathrm{Ref}_{\boldsymbol{a},\boldsymbol{c}}(\mathrm{Ref}_{\boldsymbol{a},\boldsymbol{c}}(\boldsymbol{v})). \tag{3.11}$$

Reflection reverses the location between two vectors in a Euclidean space through an affine hyperplane called a *mirror*. $\boldsymbol{a}$ and $\boldsymbol{c}$ are parameters that determine the mirror. $\boldsymbol{a} \in \mathbb{R}^n$ is a vector orthogonal to the mirror and $\boldsymbol{c} \in \mathbb{R}^n$ is a point through which the mirror passes. Reflection is different from inverse mapping. When m and w are paired words, reflection can transfer $\boldsymbol{v}_m$ and $\boldsymbol{v}_w$ between each

other with identical reflection mapping as shown in Eqs. 3.5 and 3.6; however, an inverse mapping cannot perform this action. Given a vector $\boldsymbol{v}$ in the Euclidean space $\mathbb{R}^n$, the formula for the reflection in the mirror is given by

$$\text{Ref}_{\boldsymbol{a},\boldsymbol{c}}(\boldsymbol{v}) = \boldsymbol{v} - 2\frac{(\boldsymbol{v}-\boldsymbol{c})\cdot\boldsymbol{a}}{\boldsymbol{a}\cdot\boldsymbol{a}}\boldsymbol{a}. \tag{3.12}$$

3.4.3 Reflection-based Word Attribute Transfer

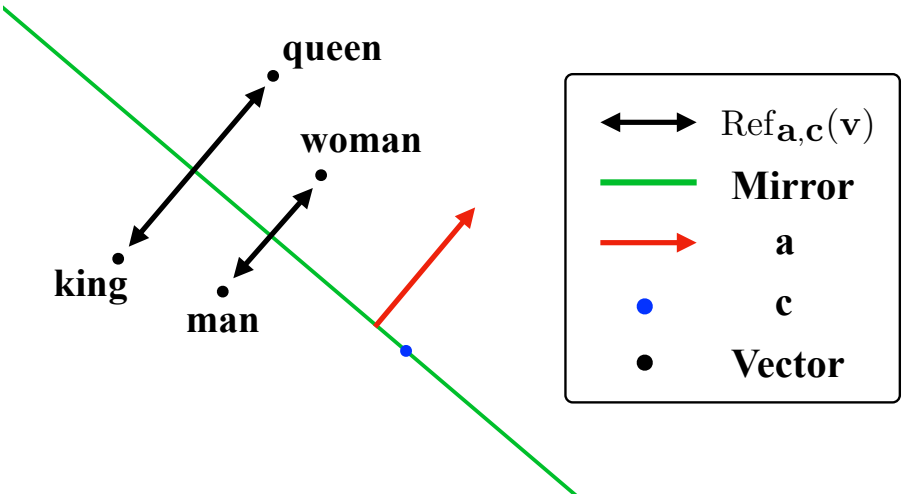


Figure 3.2: Reflection-based word attribute transfer with a single mirror

Reflection by a Single Mirror We apply reflection to the word attribute transfer process. We learn a mirror (hyperplane) in a pretrained embedding space using training word pairs with binary attribute $\boldsymbol{z}$ (Figure. 3.2). Because the mirror is uniquely determined by two parameter vectors, $\boldsymbol{a}$ and $\boldsymbol{c}$, we estimate $\boldsymbol{a}$ and $\boldsymbol{c}$ from the target attribute $\boldsymbol{z}$ using fully connected multilayer perceptrons (MLPs):

$$\boldsymbol{a} = \text{MLP}_{\theta_1}(\boldsymbol{z}), \tag{3.13}$$

$$\boldsymbol{c} = \text{MLP}_{\theta_2}(\boldsymbol{z}), \tag{3.14}$$

where θ is a set of trainable parameters of MLP_θ. The transferred vector $\boldsymbol{v}_y$ is obtained by inverting the attribute $\boldsymbol{z}$ of $\boldsymbol{v}_x$ by reflection:

$$\boldsymbol{v}_y = \text{Ref}_{\boldsymbol{a},\boldsymbol{c}}(\boldsymbol{v}_x). \tag{3.15}$$

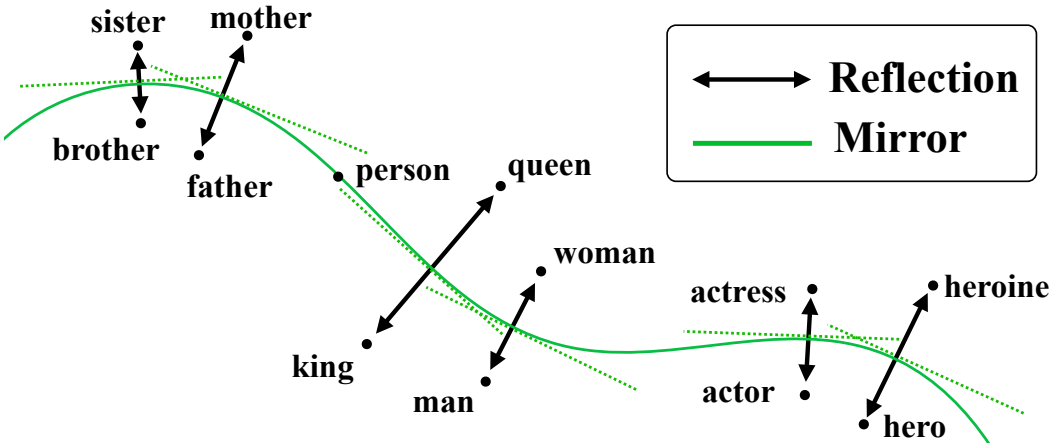


Figure 3.3: Reflection using parameterized mirrors

Reflection by Parameterized Mirrors Reflection with a mirror according to Eqs. 3.13 and 3.14 assumes a single mirror that only depends on $\boldsymbol{z}$. The previous discussion assumed pairs that share a stable pair, such as *king* and *queen*.

However, as gender-variant words often do not come in pairs, gender is not sufficiently stable to be modeled by a single mirror. For example, although *actress* is exclusively feminine, *actor* is clearly neutral in several cases. Thus, *actor* is not a masculine counterpart such as *king*. In fact, bias exists in gender words in the embedding space [54, 114]. This phenomenon can occur not only with gender attributes but also with other attributes. The assumption of a single mirror forces the mirror to be a hyperplane that goes through the midpoints for all word vector pairs. However, the vector pair *actor*-*actress*, shown on the right in Figure. 3.3, cannot be transferred well as the single mirror (the green line) does not satisfy this constraint owing to the bias of the embedding space. To solve this problem, we propose *parameterized mirrors* based on the idea of using different mirrors for different words. We define the mirror parameters $\boldsymbol{a}$ and $\boldsymbol{c}$ using the word vector $\boldsymbol{v}_x$ to be transferred in addition to the attribute vector $\boldsymbol{z}$:

$$\boldsymbol{a} = \mathrm{MLP}_{\theta_1}([\boldsymbol{z}; \boldsymbol{v}_x]), \tag{3.16}$$

$$\boldsymbol{c} = \mathrm{MLP}_{\theta_2}([\boldsymbol{z}; \boldsymbol{v}_x]), \tag{3.17}$$

where $[\cdot;\cdot]$ indicates the vector concatenation in the column.

Parameterized mirrors are expected to work more flexibly on different words than a single mirror because *parameterized mirrors* dynamically determine similar

Algorithm 1 Parameterized mirrors at inference

Input: Input word x

Input: Transfer target attribute $\boldsymbol{z}$

Output: Output word y

$\boldsymbol{v}_x \leftarrow$ PRE_TRAINED_WORD_EMBEDDING(x) ▷ Get the word vector

// Generate a mirror parameters dynamically according to the input word vector $\boldsymbol{v}_x$ and an attribute $\boldsymbol{z}$.

$\boldsymbol{a} \leftarrow \mathrm{MLP}_{\theta_1}([\boldsymbol{z}; \boldsymbol{v}_x])$ ▷ Generate a mirror parameter (**mirror direction vector**)

$\boldsymbol{c} \leftarrow \mathrm{MLP}_{\theta_2}([\boldsymbol{z}; \boldsymbol{v}_x])$ ▷ Generate a mirror parameter (**mirror position vector**)

// Generate a mirror based on $\boldsymbol{a}$ and $\boldsymbol{c}$ and changing the meaning of words using reflection with a parameterized mirror.

$\boldsymbol{v}_y \leftarrow \mathrm{Ref}(\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{v}_x)$

mirrors for similar words. Note that the number of mirrors is not predetermined, so the operation of selecting the best mirror from several mirrors is not performed. Instead, the mirror is automatically generated from the input vector (Algorithm 1). The mirror is uniquely determined by two mirror parameter vectors. In the parameterized mirror, these parameters are learned from the input word vector. As a result, when a word vector for the test and an attribute vector is input during the test, the MLPs can generate the mirror that transforms the attribute of the word vector. For example, let's assume that the reflection is learned to convert *hero* to *heroine* as shown in Figure 3.3. Now the MLPs can generate the mirror parameter vectors to convert *hero* to *heroine*. Suppose that a vector similar to *hero*, such as *actor*, is input during the test. Here, *hero* and *actor* are similar vectors (the neighbourhood in terms of Euclidean distance), so the mirror parameter vectors generated by the MLPs for *actor* should be similar to the one for *hero*. An important thing is that the mirror is generated dynamically according to the input word vector, resulting in a mirror that optimally reflects the features of the input word. Therefore, even for words that have not been learned, the parameterized mirror can function well.

It is also possible to learn reflection to not transform attributes-invariant words, such as *person* in Fig 3.3). This can be done by training the reflection to output

the same word vector when inputting such word vectors. The reflection works as an identity mapping for a vector on the mirror.

It should be noted that Eq. 3.11 may not hold for parameterized mirrors. In the reflection with a single mirror, it is true that $\boldsymbol{v} = \mathrm{Ref}_{\boldsymbol{a},\boldsymbol{c}}(\mathrm{Ref}_{\boldsymbol{a},\boldsymbol{c}}(\boldsymbol{v}))$. However, this is not guaranteed with the $\boldsymbol{v}$-parameterized reflection $\mathrm{Ref}_{\boldsymbol{a}_v,\boldsymbol{c}_v}(\boldsymbol{v})$. This is because the mirror parameters $\boldsymbol{a}_v$ and $\boldsymbol{c}_v$ depend on an input word vector, as shown in Eqs. 3.16 and 3.17. Thus, we exclude this constraint and employ the constraints given by Eqs. 3.5-3.7 for our loss function.

Weight Sharing In neural networks, weight sharing can reduce the number of trainable weights and often improve performance [61,109]. The mirror parameters $\boldsymbol{a}$ and $\boldsymbol{c}$ can be defined using a shared MLP as follows:

$$\boldsymbol{o} = \mathrm{MLP}_\theta([\boldsymbol{z}; \boldsymbol{v}_x]), \tag{3.18}$$

$$\boldsymbol{a} = \boldsymbol{W}_a \boldsymbol{o}, \tag{3.19}$$

$$\boldsymbol{c} = \boldsymbol{W}_c \boldsymbol{o}, \tag{3.20}$$

where θ indicates the shared weights. $\boldsymbol{o} \in \mathbb{R}^m$ is an output vector of MLP_θ. $\boldsymbol{W}_a \in \mathbb{R}^{n \times m}$ and $\boldsymbol{W}_c \in \mathbb{R}^{n \times m}$ are weight matrices corresponding to $\boldsymbol{a}$ and $\boldsymbol{c}$, respectively.

Loss The following properties must be satisfied in word attribute transfer: (1) words with attribute $\boldsymbol{z}$ are transferred and (2) words without it are not transferred. Thus, loss $L(\Theta)$ is defined as:

$$L(\Theta) = \frac{1}{|\mathrm{A}|} \sum_{(x,t,\boldsymbol{z}) \in \mathrm{A}} (\boldsymbol{v}_y - \boldsymbol{v}_t)^2 + \frac{1}{|\mathrm{N}|} \sum_{(x,\boldsymbol{z}) \in \mathrm{N}} (\boldsymbol{v}_y - \boldsymbol{v}_x)^2, , \tag{3.21}$$

where Θ is a set of trainable parameters ($\Theta = \{\theta\}$ for weight sharing and $\Theta = \{\theta_1, \theta_2\}$ otherwise). The first term draws the target word vector $\boldsymbol{v}_{t_i}$ closer to the corresponding transferred vector $\boldsymbol{v}_{y_i}$ and the second term prevents words that are invariant with respect to a target attribute from being moved by the transfer function. $\boldsymbol{v}_y$ is the output of a reflection (Eq. 3.15).

3.5 Experiments

We evaluated the performance of word attribute transfer using data with four different attributes. We used 300-dimensional word2vec [†] and GloVe [‡] models as the pretrained word embedding. We used four different datasets of word pairs with four binary attributes: Male-Female (MF), Singular-Plural (SP), Capital-Country (CC), and Antonym (AN) (Table 3.1). These word pairs were collected from analogy test sets [39,74] and the Internet. Antonyms were obtained from the literature [79]. Their datasets were collected from WordNet [77] and Wordnik [§]. The original data by Nguyen et al. [79] contains synonyms; however, we excluded them and used only the antonyms. We compared the models that train with attributes individually with the models that train with joint attributes. The invariant word dataset N were constructed by random sampling from WordNet by excluding the attribute-variant words in the corresponding set A. We sampled the invariant words for the invariant portion of the training data by varying their occupancy, i.e., 0, 5, 10, 25, and 50%, to investigate their effects on the tradeoffs between variant and invariant words. We also chose 1,000 invariant words for the test ($|N_{test}|$ =1,000).

Table 3.1: Statistics of binary-attribute word datasets

Dataset A	**#Train**	**#Val**	**#Test**	**#Total**
Male-Female (MF)	106	48	48	202
Singular-Plural (SP)	3624	776	776	5176
Capital-Country (CC)	118	50	50	218
Antonym (AN)	5002	642	642	6286

3.5.1 Evaluation Metrics

We measured the accuracy and stability performances of the word attribute transfer. The accuracy measures the number of input words in A_{test} that were trans-

[†] https://code.google.com/archive/p/word2vec/
[‡] https://nlp.stanford.edu/projects/glove/
[§] https://www.wordnik.com/

ferred correctly to the corresponding target words. The stability score measures the number of words in $\mathrm{N}_{\mathrm{test}}$ that were not mapped to other words. For example, in the MF transfer, given *man*, the transfer is regarded as correct if *woman* is the closest word to the transferred vector; otherwise, it is incorrect. Given *person*, the transfer is regarded as correct if *person* is the closest word to the transferred vector; otherwise, it is incorrect. The accuracy and stability scores are calculated using the following formula:

$$\delta(\boldsymbol{v}_y, t) = \begin{cases} 1 & \text{if} \quad \arg\max_{k \in \mathrm{V}}(\cos(\boldsymbol{v}_y, \boldsymbol{v}_k)) = t \\ 0 & \text{otherwise,} \end{cases} \tag{3.22}$$

$$\tag{3.23}$$

$$\text{Accuracy} = \frac{1}{|\mathrm{A}_{\mathrm{test}}|} \sum_{(x,t,\boldsymbol{z}) \in \mathrm{A}_{\mathrm{test}}} \delta(\boldsymbol{v}_y, t), \tag{3.24}$$

$$\text{Stability} = \frac{1}{|\mathrm{N}_{\mathrm{test}}|} \sum_{(x,\boldsymbol{z}) \in \mathrm{N}_{\mathrm{test}}} \delta(\boldsymbol{v}_y, x), \tag{3.25}$$

where V is the vocabulary of the word embedding model and $\cos(\boldsymbol{v}_y, \boldsymbol{v}_k)$ is the cosine similarity measure, which is defined as $\cos(\boldsymbol{v}_y, \boldsymbol{v}_k) = \frac{\boldsymbol{v}_y \cdot \boldsymbol{v}_k}{\|\boldsymbol{v}_y\| \|\boldsymbol{v}_k\|}$.

For the accuracy evaluation in the AN transfer, we used a different definition, as presented subsequently, to evaluate the accuracy because there are multiple possible candidates for the transfer in the AN dataset.

$$\delta_{\mathrm{AN}}(\boldsymbol{v}_y, t) = \begin{cases} 1 & \text{if} \quad \arg\max_{k \in \mathrm{V}}(\cos(\boldsymbol{v}_y, \boldsymbol{v}_k)) \in \mathrm{T}, \\ 0 & \text{otherwise,} \end{cases} \tag{3.26}$$

$$\tag{3.27}$$

$$\text{Accuracy}_{\mathrm{AN}} = \frac{1}{|\mathrm{A}_{\mathrm{test}}|} \sum_{(x,\mathrm{T},\boldsymbol{z}) \in \mathrm{A}_{\mathrm{test}}} \delta_{\mathrm{AN}}(\boldsymbol{v}_y, \mathrm{T}), \tag{3.28}$$

where $\mathrm{T} = \{t_1, t_2, ..., t_3\}$ is a set of target words of the input antonym word x.

3.5.2 Methods and Configurations

Because these datasets are significantly small, we added 300-dimensional Gaussian noise to every input vector during training to avoid overfitting, i.e., $\boldsymbol{v}_x + \boldsymbol{g}_\sigma$, where $\boldsymbol{g}_\sigma$ is the Gaussian noise and σ is the standard deviation of the Gaussian distribution. The reported results are presented for the best set of hyperparameters evaluated on the validation set for each model after a grid search on the following values: Adam [57] learning rate $\alpha \in \{0.0001, 0.00015, 0.001, 0.0015\}$ (the other hyperparameters were the same as the original hyperparameters), $\sigma \in \{0.0, 0.05, 0.1, 0.15, 0.2\}$, and MLP inner hidden size $\in \{300, 500, 1500, 3000\}$. Table 3.2 lists the best hyperparameters of the proposed method. We did not use regularization methods such as dropout [92] or batch normalization [47] because they did not show any improvement in our pilot test.

In training, the attribute and invariant word data were combined into one training dataset, where an invariant word $(x, \boldsymbol{z}) \in \mathrm{N}$ was represented as $(x, x, \boldsymbol{z}) \in \mathrm{N}$, similar to an attribute word $(x, t, \boldsymbol{z}) \in \mathrm{A}$. Thus, we could simply implement the loss function (Eq. 3.21) as follows: $L(\Theta) = \frac{1}{|\mathrm{A} \cap \mathrm{N}|} \sum_{(x,t,\boldsymbol{z}) \in \mathrm{A} \cap \mathrm{N}} (\boldsymbol{v}_y - \boldsymbol{v}_t)^2$, where $\mathrm{A} \cap \mathrm{N}$ is a mini-batch of training data.

In our experiment, we compared our proposed method with the following baseline methods:

Ref This is a reflection-based word attribute transfer with a single mirror. We used a fully connected MLP with a rectified linear unit (ReLU) [40] to estimate $\boldsymbol{a}$ and $\boldsymbol{c}$.

Ref+PM This is a reflection-based word attribute transfer with *parameterized mirrors*. We used the same architecture of MLP as Ref.

Ref+PM+Share This method consists of a reflection-based word attribute transfer with *parameterized mirrors*. We used the MLP with shared weights to estimate $\boldsymbol{a}$ and $\boldsymbol{c}$ (refer to Weight Sharing in Section 3.4.3).

MLP This is a fully connected MLP with ReLU: $\boldsymbol{v}_y = \mathrm{MLP}([\boldsymbol{v}_x; \boldsymbol{z}])$. The highest accuracy models are a five-layer MLP with 1500 hidden units for SP, five-layer MLP with 3000 hidden units for AN, and three-layer MLP with 300 hidden units for the other datasets. The optimal configurations were as

follows: the learning rate for Adam $\alpha = 0.00015$ for all datasets; moreover, $\sigma = 0.05$ for AN and $\sigma = 0.1$ for the other datasets.

TransE The word attribute transfer task is similar to link prediction in which (x, z, t) is replaced with $(\text{head}, \text{label}, \text{tail})$. In link prediction, given a set of triplets $(\text{head}, \text{label}, \text{tail})$, the knowledge graph embedding model predicts the tail from the head and label. We applied TransE [16], a baseline model for knowledge graph embeddings, to word attribute transfer. We modified the model to input the word vector into the knowledge graph embedding model based on the following equations:

$$\boldsymbol{h} = \boldsymbol{W}_{\text{head}} \boldsymbol{v}_x, \tag{3.29}$$

$$\boldsymbol{t} = \boldsymbol{W}_{\text{tail}} \boldsymbol{v}_t, \tag{3.30}$$

where $\boldsymbol{h} \in \mathbb{R}^k$ is a head vector and $\boldsymbol{t} \in \mathbb{R}^k$ is a tail vector. $\boldsymbol{W}_{\text{head} \in \mathbb{R}^{k \times n}}$ and $\boldsymbol{W}_{\text{tail} \in \mathbb{R}^{k \times n}}$ are weight matrices corresponding to the head and tail, respectively. The label vector $\boldsymbol{l}$ was embedded in the same way as the original TrasnE based on a set of relations $\{\text{MF}, \text{SP}, \text{CC}, \text{AN}\}$. The optimal configurations were as follows: the latent dimension $k = 200$, learning rate λ for stochastic gradient descent $\lambda = 1.0$, and margin $\gamma = 5.0$. TransE was implemented using an open toolkit for knowledge embedding called OpenKE [42]. In the evaluation, when calculating the accuracy and stability in Eqs. 3.23 and 3.27, the score function of TransE was used instead of $\cos(\boldsymbol{v}_y, \boldsymbol{v}_k)$.

Diff This method consists of analogy-based word attribute transfer with a difference vector, $\boldsymbol{d} = \boldsymbol{v}_m - \boldsymbol{v}_w$, where m and w are in the training data of A. We chose the $\boldsymbol{d}$ that achieved the best accuracy in the validation data of A. We determined whether to add or subtract $\boldsymbol{d}$ to $\boldsymbol{v}_x$ based on explicit knowledge (Eq. 3.4). Here, DIFF$^+$ and DIFF$^-$ transfer word attributes using a difference vector regardless of the explicit knowledge. $^+$ and $^-$ add or subtract the difference vector to any input word vector.

MeanDiff This method includes an analogy-based word attribute transfer with a mean difference vector $\bar{\boldsymbol{d}}$, where

$\bar{\boldsymbol{d}} = \frac{1}{|\mathrm{A}_{\mathrm{train}}|} \sum_{(m_i, w_i, \boldsymbol{z}) \in \mathrm{A}_{\mathrm{train}}} (\boldsymbol{v}_{m_i} - \boldsymbol{v}_{w_i})$. We determined whether to add or subtract $\bar{\boldsymbol{d}}$ to $\boldsymbol{v}_x$ based on the explicit knowledge (Eq. 3.4).

Table 3.2: Hyperparameters for reflection-based word attribute transfer

Embedding	Hyperparameters	MF	SP	CC	AN
word2vec	**Model**	Ref+PM	Ref+PM+Share	Ref+PM	Ref+PM
	Batch size	512	512	512	4096
	Best Iterations	21000	14000	20000	20000
	Noise σ	0.1	0.1	0.1	0.1
	Adam α	0.0001	0.0001	0.0001	0.0001
	Activation Function	ReLU	ReLU	ReLU	ReLU
	Num of MLP layers	3	5	3	5
	Inner hidden size of MLP	300	1500	300	3000
	Size of $\lvert \mathrm{N}_{\mathrm{train}} \rvert$	5%	25%	5%	50%
GloVe	**Model**	Ref+PM+Share	Ref+PM+Share	Ref+PM	Ref+PM
	Batch size	512	512	512	4096
	Best Iterations	48000	20000	24000	30000
	Noise σ	0.1	0.1	0.1	0.1
	Adam α	0.0001	0.0001	0.0001	0.0001
	Activation Function	ReLU	ReLU	ReLU	ReLU
	Num of MLP layers	3	5	3	5
	Inner hidden size of MLP	300	1500	300	3000
	Size of $\lvert \mathrm{N}_{\mathrm{train}} \rvert$	25%	10%	5%	10%

3.5.3 Evaluation of Accuracy and Stability

Table 3.3 lists the accuracy and stability results. Because AN has a many-to-many relationship, a single difference vector cannot be obtained. Therefore, the analogy methods (Diff, Diff^{+-}, MeanDiff, and MeanDiff^{+-}) were not applied to AN. Different pretrained word embeddings by GloVe and word2vec provided similar results. Ref+PM and Ref+PM+Share achieved the best accuracy among the methods that did not use explicit attribute knowledge. For example, the accuracy of Ref+PM was 74% for CC; however, the accuracy of MLP was 18%. For most attributes, our proposed methods outperformed the analogy-based transfer. Weight sharing did not significantly improve the performance of the proposed methods. The parameterized mirror improved the performance of a reflection-based transfer, although the learning was unstable (Figure. 3.4).

For stability, reflection-based transfers achieved superior stability scores, which exceeded 93% in most cases. We mixed all the attribute datasets and trained models. The best model was the proposed method trained with an individual attribute dataset. In the joint condition, the MLP demonstrated better performance than that in the individual attribute condition with the help of the larger training data. The results show that our proposed methods transfer an input word if it has a target attribute and does not transfer an input word with better scores than the baseline methods, even though the proposed methods do not use knowledge of the input words.

Table 3.3: Results of accuracy and stability scores. MF, SP, CC, and AN are datasets. Here, "joint" models are trained with joint attributes and "individual" models are trained with an individual attribute.

Embedding	Method	Knowledge	Accuracy (%)				Stability (%)			
			MF	**SP**	**CC**	**AN**	**MF**	**SP**	**CC**	**AN**
word2vec	REF (individual)		22.9	0.5	44.0	0.2	**100.0**	**100.0**	**100.0**	**100.0**
	REF+SHARE (individual)		22.9	0.5	42.0	0.2	**100.0**	**100.0**	**100.0**	**100.0**
	REF+PM (individual)		**41.7**	44.2	**62.0**	11.2	98.5	95.9	**100.0**	75.2
	REF+PM+SHARE (individual)		37.5	43.0	60.0	7.2	99.3	98.7	**100.0**	96.2
	MLP (individual)		10.4	40.1	18.0	**12.5**	5.7	95.1	9.2	92.4
	TransE (individual)		0.0	0.2	0.0	0.0	**100.0**	**100.0**	**100.0**	**100.0**
	REF (joint)		18.8	0.3	42.0	0.2	**100.0**	**100.0**	**100.0**	**100.0**
	REF+SHARE (joint)		18.8	0.3	44.0	0.2	**100.0**	**100.0**	**100.0**	**100.0**
	REF+PM (joint)		25.0	43.4	44.0	13.2	**100.0**	93.7	**100.0**	63.7
	REF+PM+SHARE (joint)		18.8	**50.8**	34.0	16.0	**100.0**	98.8	**100.0**	89.0
	MLP (joint)		16.7	38.1	8.0	14.0	95.4	98.6	97.6	97.1
	TransE (joint)		0.0	0.3	0.0	0.0	**100.0**	**100.0**	**100.0**	**100.0**
	DIFF $^+$		22.9	3.2	32.0	-	97.1	93.1	89.5	-
	DIFF $^-$		22.9	3.1	32.0	-	87.9	98.8	99.4	-
	MEANDIFF $^+$		6.3	0.3	22.0	-	**100.0**	**100.0**	99.7	-
	MEANDIFF $^-$		8.3	0.3	14.0	-	**100.0**	**100.0**	99.9	-
	DIFF	✓	37.5	6.3	64.0	-	-	-	-	-
	MEANDIFF	✓	14.6	0.5	36.0	-	-	-	-	-
GloVe	REF (individual)		10.4	0.5	24.0	0.2	**100.0**	**100.0**	**100.0**	**100.0**
	REF+SHARE (individual)		10.4	0.4	24.0	0.0	**100.0**	**100.0**	**100.0**	**100.0**
	REF+PM (individual)		37.5	45.0	**74.0**	12.3	99.2	99.3	**100.0**	93.3
	REF+PM+SHARE (individual)		**39.6**	42.8	72.0	11.5	99.2	99.8	**100.0**	94.7
	MLP (individual)		14.6	41.1	18.0	**14.2**	41.7	97.6	50.3	93.4
	TransE (individual)		0.0	0.2	0.0	0.0	**100.0**	**100.0**	**100.0**	**100.0**
	REF (joint)		12.5	0.4	24.0	0.2	**100.0**	**100.0**	**100.0**	**100.0**
	REF+SHARE (joint)		2.1	0.4	20.0	0.0	**100.0**	**100.0**	**100.0**	**100.0**
	REF+PM (joint)		12.5	39.8	36.0	9.0	**100.0**	99.7	**100.0**	94.1
	REF+PM+SHARE (joint)		12.5	**47.0**	36.0	9.8	**100.0**	97.7	**100.0**	59.0
	MLP (joint)		27.1	35.2	26.0	11.1	98.0	99.7	99.4	97.6
	TransE (joint)		0.0	0.3	0.0	0.0	**100.0**	**100.0**	**100.0**	**100.0**
	DIFF $^+$		14.6	4.5	22.0	-	**100.0**	**100.0**	**100.0**	-
	DIFF $^-$		12.5	4.3	26.0	-	**100.0**	99.8	99.9	-
	MEANDIFF $^+$		0.0	0.3	2.0	-	**100.0**	**100.0**	**100.0**	-
	MEANDIFF $^-$		0.0	0.3	4.0	-	**100.0**	**100.0**	**100.0**	-
	DIFF	✓	27.1	8.7	48.0	-	-	-	-	-
	MEANDIFF	✓	0.0	0.5	6.0	-	-	-	-	-

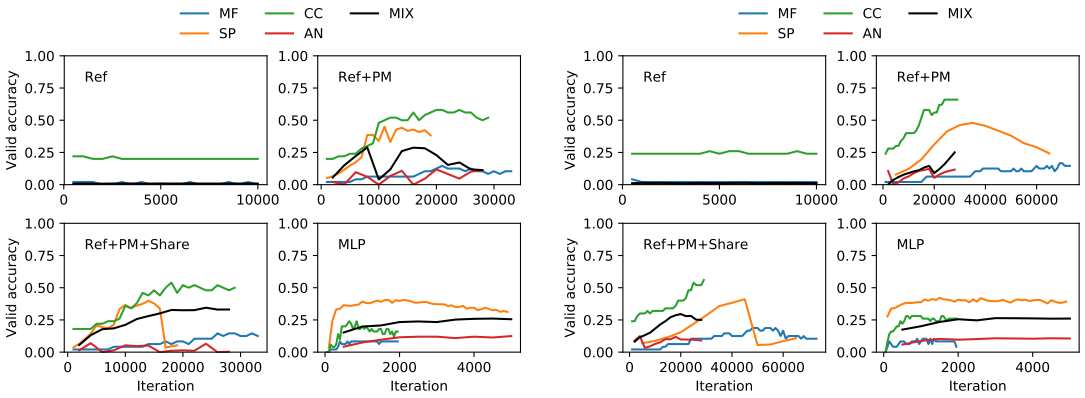
Table 3.4: Accuracy of the top three nearest neighbors of TransE. A "joint" model is trained with joint attributes. An "individual" model is trained with an individual attribute.

Method	word2vec								GloVe							
	MF		SP		CC		AN		MF		SP		CC		AN	
	@1	@3	@1	@3	@1	@3	@1	@3	@1	@3	@1	@3	@1	@3	@1	@3
TransE (individual)	0.0	75.0	0.2	76.3	0.0	80.0	0.0	20.2	0.0	66.7	0.2	80.0	0.0	76.0	0.0	42.2
TransE (joint)	0.0	72.9	0.3	70.7	0.0	64.0	0.0	20.0	0.0	64.6	0.3	75.6	0.0	72.0	0.0	36.4

In TransE, the accuracy was almost 0% for all the attributes[¶]. However, in the MF, SP, and CC relations, several reference words were within the top three nearest neighbors, as listed in Table 3.4. This result is similar not only in this task but also when learning with WN18 [97]. This is owing to the nature of TransE. While considering AN, the accuracy in the three nearest neighbors was still low. This can be explained by the poor performance of TransE for one-to-many or many-to-many relationships [16].

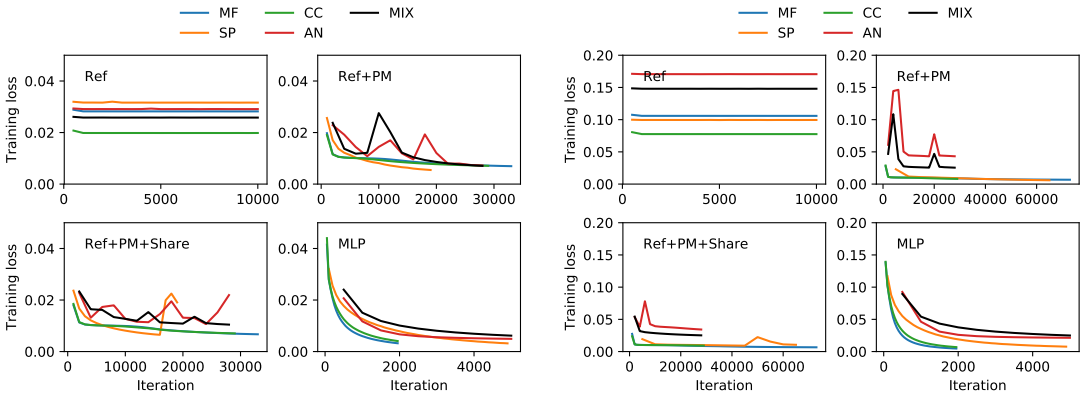
In the individual attribute condition, MLP worked poorly, especially in terms of stability for MF and CC, while it showed better transfer accuracy for AN than the proposed method. We reviewed the training curves resulting from the MLP, which are shown in Figures 3.4 and 3.5; however, they showed reasonable convergence. This would be due to the training data size in the individual attribute condition, because MLP stability significantly improves in the joint condition.

[¶]We also experimented with ComplEx [97], which is a knowledge graph embedding model in a complex space; however, it is not described in this thesis because it is not comparable because of its low accuracy.



(a) word2vec **(b)** GloVe

Figure 3.4: Visualization of validation accuracy



(a) word2vec **(b)** GloVe

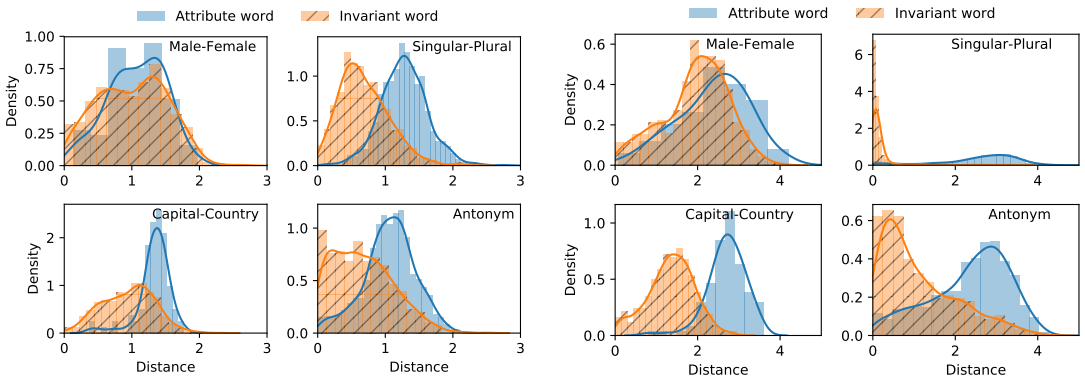
Figure 3.5: Visualization of training loss

We also investigated the tradeoff between transfer accuracy and stability by changing the size of the invariant words and the stability of the learning-based methods by conducting an additional experiment that varied $|\mathrm{N}_{\mathrm{train}}|$. The large size of $\mathrm{N}_{\mathrm{train}}$ is expected to increase the stability; however, it may also decrease the accuracy. The stability scores demonstrated by the MLP did not improve (Table 3.5) for MF and CC. Conversely, the proposed methods achieved high stability scores with $|\mathrm{N}_{\mathrm{train}}| = 5\%$ and maintained the accuracy. We hypothesized that the high stability was owing to the distance between the word and its mirror. If invariant words are distributed on the mirror, they will not be transferred. We investigated the distance between the input word vector $\boldsymbol{v}_x$ and its mirror (Figure. 3.6). The result showed that invariant words were close to the mirror and attribute words were distributed away from it. If the distance between paired words is significantly small, the distance between the word and its mirror is also small. Figure. 3.7 shows the distribution of the distance between the input $\boldsymbol{v}_x$ and the target word vector $\boldsymbol{v}_t$. The distance between paired words for MF and SP is considerably smaller than that for CC and AN.

Although analogy-based methods achieved high stability, their accuracy results were low. In particular, the MeanDiff$^+$ and MeanDiff$^-$ did not change the original vector. We hypothesized that the result can be attributed to the significantly small L2 norm of the mean difference vector $\bar{\boldsymbol{d}}$. Table 3.6 lists the relationship between the MeanDiff performances and the L2 norm of the mean

Table 3.5: Relation between the size of $|N_{train}|$ and the stability of methods trained with an individual attribute

Embedding		Method	Accuracy (%)					Stability (%)				
			$\|N_{train}\|$					$\|N_{train}\|$				
			0%	5%	10%	25%	50%	0%	5%	10%	25%	50%
word2vec	MF	Ref	18.8	20.8	22.9	22.9	18.8	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
		Ref+PM	35.4	**41.7**	**37.5**	35.4	25.0	86.5	98.5	99.6	99.7	91.8
		Ref+PM+Share	**37.5**	31.2	35.4	**37.5**	**29.2**	78.6	99.4	99.5	99.3	99.8
		MLP	4.2	6.2	8.3	8.3	10.4	0.0	0.0	0.0	0.8	5.7
	SP	Ref	0.4	0.5	0.4	0.3	0.3	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
		Ref+PM	43.3	**46.3**	**44.2**	42.4	**40.3**	53.4	82.4	95.9	99.1	99.5
		Ref+PM+Share	**44.7**	43.6	43.7	**43.0**	38.7	42.3	93.5	94.5	98.7	98.4
		MLP	42.0	41.0	40.1	36.7	36.0	66.8	86.0	95.1	98.1	99.6
	CC	Ref	34.0	34.0	36.0	38.0	44.0	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
		Ref+PM	**62.0**	**62.0**	54.0	54.0	50.0	90.0	**100.0**	**100.0**	**100.0**	99.8
		Ref+PM+Share	56.0	58.0	**58.0**	**60.0**	**56.0**	86.4	**100.0**	**100.0**	**100.0**	**100.0**
		MLP	10.0	12.0	10.0	18.0	18.0	0.0	0.0	0.0	0.6	9.2
	AN	Ref	0.0	0.2	0.0	0.0	0.2	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
		Ref+PM	12.5	12.9	12.3	11.8	11.2	26.8	26.0	34.3	65.7	75.2
		Ref+PM+Share	13.9	12.8	12.1	12.0	7.2	7.7	20.8	49.7	71.4	96.2
		MLP	**17.0**	**15.4**	**15.1**	**12.5**	**14.2**	1.2	6.2	36.6	92.4	67.2
GloVe	MF	Ref	10.4	4.2	6.2	4.2	2.1	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
		Ref+PM	**37.5**	**39.6**	37.5	37.5	**35.4**	89.3	93.2	95.7	99.2	99.6
		Ref+PM+Share	35.4	31.2	**39.6**	**39.6**	**35.4**	88.7	98.9	97.5	99.2	99.6
		MLP	4.2	12.5	6.2	8.3	14.6	0.0	0.0	0.0	0.3	41.7
	SP	Ref	0.5	0.4	0.5	0.4	0.4	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
		Ref+PM	**46.3**	**46.6**	**46.4**	44.6	**45.0**	54.1	94.5	97.8	98.9	99.3
		Ref+PM+Share	43.9	43.7	44.8	**45.0**	42.8	52.6	95.5	98.1	99.4	99.8
		MLP	42.7	41.0	41.1	38.9	36.9	70.0	95.2	97.6	99.3	99.8
	CC	Ref	22.0	24.0	24.0	22.0	20.0	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
		Ref+PM	66.0	**74.0**	**70.0**	70.0	**74.0**	99.9	**100.0**	99.9	**100.0**	99.9
		Ref+PM+Share	**70.0**	70.0	**70.0**	**72.0**	72.0	99.8	99.9	99.9	99.9	**100.0**
		MLP	8.0	6.0	8.0	6.0	18.0	0.0	0.0	0.0	1.3	50.3
	AN	Ref	0.2	0.2	0.0	0.0	0.0	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
		Ref+PM	12.1	13.2	12.3	10.9	10.6	16.0	78.0	93.3	96.1	97.1
		Ref+PM+Share	11.7	12.5	10.7	11.5	8.4	14.0	42.4	73.9	94.7	96.8
		MLP	**16.6**	**14.5**	**16.0**	**14.2**	**11.7**	2.1	53.1	70.5	93.4	97.9



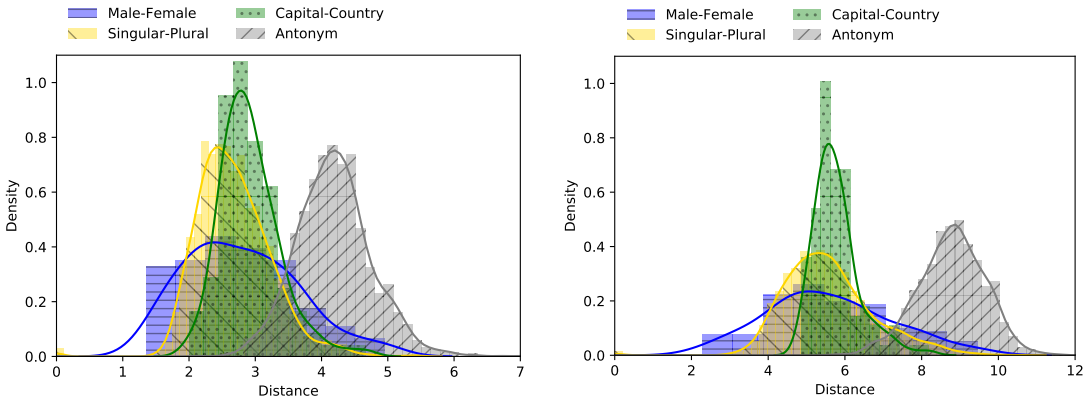
(a) word2vec **(b)** GloVe

Figure 3.6: Distribution of distance between the input word vector and its mirror $\frac{|(\boldsymbol{v}_x - \boldsymbol{c}) \cdot \boldsymbol{a}|}{\|\boldsymbol{a}\|}$ learned by REF+PM. It can be observed that invariant words are close to the mirror and attribute words are distributed away from it.

difference vector. The stability was high because the original vector was almost unchanged even if $\boldsymbol{d}$ was added or subtracted. Conversely, when the L2 norm of $\bar{\boldsymbol{d}}$ was large, the accuracy became high as the difference vectors were similar to each other.

DIFF, DIFF$^+$, and DIFF$^-$ obtained high accuracy for CC and low accuracy for SP. This is due to the use of a fixed difference vector $\boldsymbol{d}$ in Diff. We investigated the mean cosine similarity between the difference vector $\boldsymbol{d}$ and other difference vectors, i.e., $\frac{1}{|\mathrm{A}|} \sum_{(x,t,\boldsymbol{z}) \in \mathrm{A}} \mathrm{cos\_similarity}(\boldsymbol{d}_{(x,t)}, \boldsymbol{d})$, where $\boldsymbol{d}_{(x,t)}$ is the difference vector of a word pair (x, t) other than $\boldsymbol{d}$. We found that the mean cosine similarity between SP words was almost 0% in DIFF$^-$, as listed in Table 3.6. Thus, when we use a single difference vector, several SP words are transferred into inappropriate words.

We evaluated the stability of the results under different random seeds. First, we calculated the validation set accuracy by changing the random seed of the reflection with parameterized mirrors 10 times. In this experiment, the parameters of the MLP were not shared, and each attribute transformation was learned individually. Table 3.7 shows the standard deviation of the accuracy. The results showed that reflection with parameterized mirrors is not very sensitive to random seed changes. In the results for each attribute, We found that MF and CC are



(a) word2vec **(b)** GloVe

Figure 3.7: Distribution of distance between the input word vector and the target word vector $\|\boldsymbol{v}_x - \boldsymbol{v}_t\|$

more sensitive than the others. This is probably due to the size of the training dataset (Table 3.1). Since the dataset size for MF and CC is significantly smaller than the other attributes, it is likely that they became more sensitive to the seed. This result suggests that increasing the number of training data leads to more stable performance.

3.5.4 Visualization of Parameterized Mirrors

Figure. 3.8 shows the principal component analysis (PCA) results of the mirror parameter $\boldsymbol{a}$ obtained for the test words. We normalized the L2 norm of $\boldsymbol{a}$ to 1 ($\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$). We compared the PCA results with the results of the model trained with the joint attributes and the model trained with an individual attribute. Similar results were obtained for both conditions. Figure. 3.8 suggests that the mirror parameters of the paired words are similar to each other and that those with an attribute form a cluster; words with the same attribute have similar mirror parameters, i.e., $\boldsymbol{a}$.

3.5.5 Transfer Example

Table 3.8 lists the gender transfer results for a tiny example sentence. Here, the attribute transfer was applied to every word in the sentence $X = \{x_1, x_2, ...\}$.

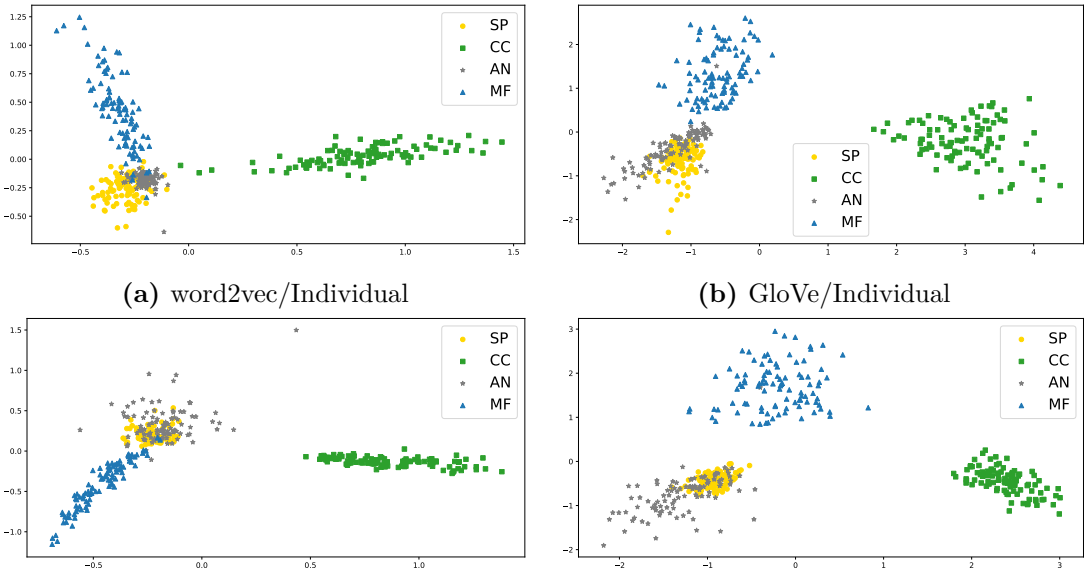
Table 3.6: Analysis of difference vectors. **L2** is the L2 norm of the difference vector that the model used during the inference time ($\boldsymbol{d}$ for DIFF and $\bar{\boldsymbol{d}}$ for MEAN-DIFF). **cos** is the distribution of cosine similarities between the difference vector ($\boldsymbol{d}$ or $\bar{\boldsymbol{d}}$) and other difference vectors.

Embedding	Attr	MEANDIFF$^-$					DIFF$^-$				
		Acc	Stb	L2	cos		Acc	Stb	L2	cos	
					mean	var				mean	var
word2vec	**MF**	8.3	100.0	1.17	0.39	0.05	22.9	97.1	2.91	0.28	0.05
	SP	0.3	100.0	0.75	0.18	0.01	3.1	98.8	3.07	0.06	0.01
	CC	14.0	99.9	1.82	0.61	0.02	32.0	99.4	2.76	0.49	0.02
GloVe	**MF**	0.0	100.0	2.29	0.38	0.03	14.6	100.0	4.61	0.29	0.04
	SP	0.3	100.0	1.82	0.21	0.02	4.5	100.0	5.68	0.08	0.01
	CC	4.0	100.0	3.56	0.61	0.02	22.0	100.0	6.05	0.44	0.01

Here, because words such as *a* and *.* are not in the vocabulary of word2vec, we omitted them from the inputs. The MLP resulted in several incorrect transfers on gender-invariant words, e.g., *the* became *dear_madam*, *were* became *laundresses*, *boyfriend* became *boyfriend*, and *boy* became *mother*. Analogy-based transfers can transfer only in one direction. DIFF$^+$ could transfer if x is female, e.g., it transferred from *woman* to *man*, but it could not transfer *boy*. Similarly, DIFF$^-$ failed to transfer from female to male. Moreover, as the stability of these methods was low, they resulted in erroneous transfers. For example, in DIFF$^-$, *the* became

Table 3.7: Standard deviation of validation accuracy when training for 10 times with changing the random seed. The model is reflection with parameterized mirrors (REF+PM). We used word2vec for the pre-trained embedding

Attribute	σ
MF	4.50
SP	1.85
CC	4.20
AN	1.57



(a) word2vec/Individual **(b)** GloVe/Individual

(c) word2vec/Joint **(d)** GloVe/Joint

Figure 3.8: Two-dimensional principal component analysis projection of the 300-dimensional mirror parameter $\boldsymbol{a}$. The mirror parameters were estimated by the proposed model (Ref+PM+Share) trained by each attribute.

Sir. Ref+PM could transfer only gender words without using explicit gender information. Thus, *woman* was transferred to *man* without the knowledge that *woman* is a female word. When the gender-invariant word *married* was input, it was not changed by reflection, without the knowledge that *married* has no gender attribute.

From Table 3.9, it can be noted that words with different target attributes were transferred by each reflection-based transfer. Thus, when *daughter* was input for a MF transfer, it was transferred to *son* and *daughters* for SP transfer. When *Tokyo* was input for MF, SF, and AN tranfers, it was not transferred; however, it was transferred to *Japan* in the CC transfer. When *stereo* was input for MF, SP, and CC transfers, it was not transferred; however, it was transferred to *monaural* in the AN transfer.

Table 3.8: Comparison of gender transfers. Each method transfers words in a sentence one by one.

	Input words	the woman got married when you were a boy.
Methods	REF	the **man** got married when you were a **boy**.
	REF+PM	the **man** got married when you were a **girl**.
	REF+PM+SHARE	the **man** got married when you were a **girl**.
	DIFF $^{+}$	Sir **man** got married when you were a **boy**.
	DIFF $^{-}$	chairwoman **woman** got married chairwoman you were a **girl**.
	MLP	dear_madam **man** dear_madam boyfriend dear_madam lazy_slob laundresses a **mother**.

3.5.6 Discussion

In our method, the performance of GloVe was better than that of word2vec. Table 3.10 lists the scores of the Google analogy test set [74] for the different embedding methods, i.e., word2vec and GloVe. From in Table 3.10, it can be noted that the score of GloVe was higher than that of word2vec. This indicates that the embedding space of GloVe worked better than that of word2vec in this task. The performance of the word attribute transfer using reflection probably depends on the analogic space, because transferring will be easy if the pair of transferred words exists in a similar place in the analogic space.

3.5.7 Error Analysis

We analyzed the attribute words that could not be transferred accurately by models trained with the individual attributes. We categorized the failed output words into three error cases.

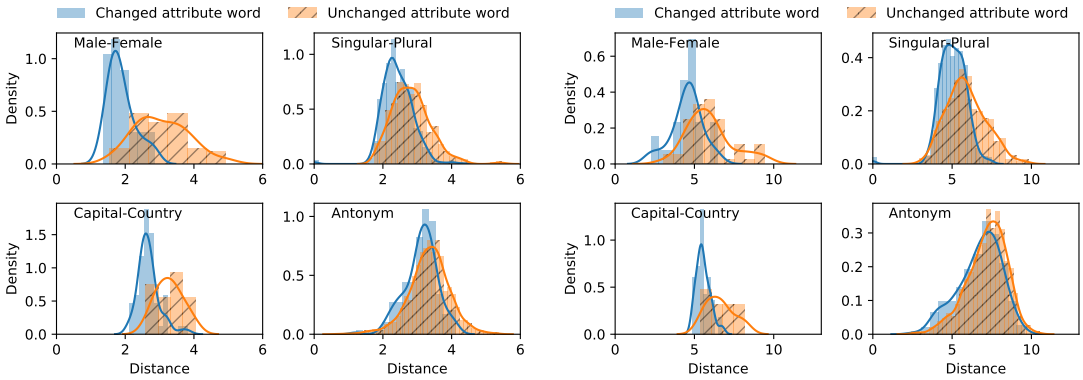
Case1 The output word was the same as the input word ($y = x$).

Case2 The attribute of the input word was transferred but was incorrect. For example, in CC transfer, the transfer result was *Beijing* when *Japan* was

given.

Case3 Other types of errors.

Table 3.11 lists the results of the error analysis in word2vec. The results show that most of the failures in the reflection-based transfer was in Case1. We speculated that such unchanged attribute word pairs tended to be close to each other. Figure. 3.9 shows the difference in the distance between the input word and the target word $\|\boldsymbol{v}_x - \boldsymbol{v}_t\|$ in the changed and unchanged attribute word pairs. Contrary to this hypothesis, it was shown that there was no difference in the distance between the changed and unchanged pairs. Table 3.12 lists examples of Case2 and Case3 in the proposed method. For example, when given *stepbrother* as a gender word, the proposed method (REF+PM) did not output *stepsister* but provided the result of *stepmother*. In MF and CC transfers, the maximum failures using MLP were observed in Case2 and Case3 errors, while the proposed methods demonstrated significant failure in the Case1 category. This shows that the reflection-based transfer is more stable in MF and CC transfers than MLP.



(a) word2vec **(b)** GloVe

Figure 3.9: Distribution of the distance between the input word vector $\boldsymbol{v}_x$ and the target word vector $\boldsymbol{v}_t$ (Comparisons between the changed and unchanged attribute words)

Table 3.9: Transfer of different attributes with the proposed method (REF+PM)

Input words		Mr. Smith and his daughter want to visit the science museum in Tokyo to see the stereo microphone .
REF+PM **(individual)**	**MF**	**Ms.** Smith and **her son** want to visit the science museum in Tokyo to see the stereo microphone.
	SP	Mr. Smith and his **daughters** want to **visits** the science **museums** in Tokyo to see the stereo **microphones**.
	CC	Mr. Smith and his daughter want to visit the science museum in **Japan** to see the stereo microphone.
	AN	Mr. Smith and his daughter **eliminate** to visit the science museum in Tokyo to **back** the **monaural** microphone.
REF+PM **(joint)**	**MF**	**Ms.** Smith and **her son** want to visit the science museum in Tokyo to see the stereo microphone.
	SP	Mr. Smith and his **daughters dos** to **visits** the science **museums** in Tokyo to **watches** the **cassette_decks microphones**.
	CC	Mr. Smith and his daughter want to visit the science museum in **Japan** to see the stereo microphone.
	AN	Mr. Smith and his daughter want to visit the **zoology** museum in Tokyo to see the **monaural** microphone.

Table 3.10: Comparison of analogy scores

Embedding method	**Analogy score (%)**
word2vec	74.01
GloVe	**75.13**

Table 3.11: Error analysis results

		Cases where transfer failed (%)		
		Case1	**Case2**	**Case3**
MF	Ref	100.0	0.0	0.0
	Ref+PM	86.0	12.0	2.0
	Ref+PM+Share	88.0	8.0	4.0
	MLP	6.0	70.0	24.0
SP	Ref	100.0	0.0	0.0
	Ref+PM	64.0	36.0	0.0
	Ref+PM+Share	62.0	38.0	0.0
	MLP	64.0	34.0	2.0
CC	Ref	100.0	0.0	0.0
	Ref+PM	37.5	62.5	0.0
	Ref+PM+Share	55.8	44.2	0.0
	MLP	2.0	98.0	0.0
AN	Ref	100.0	0.0	0.0
	Ref+PM	42.0	10.0	48.0
	Ref+PM+Share	94.0	2.0	4.0
	MLP	66.0	10.0	24.0

Table 3.12: Examples of Case2 and Case3 errors in the proposed method

Attribute	Error type	Input (x)	Target (t)	Output (y)
MF	Case2	hens	roosters	oxen
	Case2	stepbrother	stepsister	stepmother
	Case2	emperor	empress	goddess
	Case3	mare	stallion	gelding
SP	Case2	killers	killer	murderer
	Case2	atoll	atolls	islands
	Case2	gulls	gull	heron
	Case2	windmill	windmills	paddlewheels
	Case2	saxophone	saxophones	trombones
	Case2	trails	trail	trailhead
	Case2	spectaculars	spectacular	extravaganza
	Case2	visa	visas	passports
	Case2	neckties	necktie	jacket
	Case2	wagons	wagon	tractor
	Case2	outlook	outlooks	forecasts
CC	Case2	Australia	Canberra	Sydney
	Case2	Canada	Ottawa	Montreal
	Case2	Jamaica	Kingston	Belmopan
	Case2	London	England	Britain
	Case2	Hungary	Budapest	Bucharest
AN	Case2	underbid	overbid	overcharged
	Case2	perfection	imperfection	imperfect
	Case2	rely	suspect, distrust	independent
	Case2	penalty	advantage, reward	acquittal
	Case2	sane	insane, crazy	irrational
	Case3	disinherit	leave, will, bequeath	disinheriting
	Case3	elder	junior	niece
	Case3	unimpressive	impressive	solid
	Case3	unhelpful	helpful	sensible
	Case3	starve	give, feed	encourage
	Case3	extraneous	intrinsic	necessary
	Case3	harmless	harmful	important

3.6 Related Work

The embedded vectors obtained by SGNS [74, 75] and GloVe [84] have analogic relations. The theory of analogic relations in word embeddings has been widely discussed [6,7,31,38,64,68]. Levy et al. [64] explain that SGNS factorizes a shifted PMI matrix. Allen et al. [6] and Ethayarajh et al. [31] argued that they proved the existence of such analogic relations without strong assumptions. In our work, we focused on the analogic relations in a word embedding space and propose a novel framework to obtain a transferred word vector with the target attribute.

Link prediction in knowledge graph embeddings can also be applied to word transfer tasks. In knowledge graph embeddings [16, 81, 97], given a set of triplets (head, tail, label), tail is predicted from head and label TransE [16] is a knowledge graph embedding model. TransE embeds entity and relation into an embedding space using a score function. The score function models the analogy-like operation translating from the head entity to the tail entity according to the relation. Trouillon et al. [97] proposed the knowledge graph embedding model based on complex values for link prediction. Their knowledge graph embedding model is better suited for modeling a variety of binary relations, including symmetric and asymmetric relations. Our task differs from link prediction in that when an invariant word for attribute $\boldsymbol{z}$ is entered, the model returns the input x.

The style transfer task presented in previous studies [49, 71] resembles ours. In style transfer, the text style of the input sentences is changed. For instance, Jain et al. [49] transferred the style from formal to informal sentences. Logeswaran et al. [71] transferred sentences by controlling attributes such as mood and tense. These style transfer tasks use sentence pairs. Our word attribute transfer task uses word pairs. Style transfer changes sentence styles; however, our task changes the word attributes.

Soricut et al. [91] studied morphological transformation based on character information. Our work aims for a more general attribute transfer, such as gender transfer and obtaining the antonym, and is not limited to morphological transformation.

3.7 Conclusion

This work aimed to transfer word binary attributes (e.g., gender) for applications such as data augmentation of a sentence[‖]. We can transfer word attributes using the analogy of word vectors; however, this process requires explicit knowledge on whether the input word has the attribute or not. However, this knowledge cannot be developed for various words and attributes in practice. The proposed method uses reflection-based mappings to transfer attribute-variant words into their counterparts while ensuring that attribute-invariant words are unchanged, without using attribute knowledge in the inference time. The experimental results showed that the proposed method outperformed baseline methods in terms of transfer accuracy for attribute-variant words and stability for attribute-invariant words. We speculated that the reason why the proposed method achieved significantly high stability was that invariant words were distributed in the mirrors. We examined the distance between the input word vector and its mirror. The result showed that invariant words were distributed near the mirror and attribute words were distributed away from the mirror.

[‖]Our code and datasets are available at: `https://github.com/ahclab/reflection`

4 Subspace-based Set Operations

4.1 Background

Word embedding is an essential technology that supports recent NLP. Such static word representations as word2vec [75] and such dynamic word representations as BERT [30] have greatly boosted the performance of various NLP tasks [27, 103, 104].

Word embedding provides a vector representation for *a word* or a token. Here we must also emphasize the importance of representing *a set of words, not just a single word.* In NLP, the object of representation for meaning is often a set of words, so the representation of a word set has many potential applications. For example, the words in set $\{red, blue, green, \dots\}$ share the concept of *color* [77]. Computation on such a word set is considered necessary in NLP [111]. Furthermore, the typical use of such word sets is the representation of text chunks, including phrases, sentences, and documents. This strategy looks very simple and usually works effectively in practice. For example, a standard principle for computing sentence similarity [4] is to calculate the degree of overlap among word sets [94].

The most classical way of dealing with a word set is by treating words as discrete symbols. We can exploit attractive set operations, such as union ($\cup$), intersection ($\cap$), set membership ($\in$), and set similarity, including Jaccard index [48]. A critical limitation of this symbolic set representation is that it cannot handle the semantic similarity of such words as *man* and *king*. Furthermore, a symbolic word set ignores word order information, which must be preserved when using word sets. As a result, different expressions are treated completely independently from each other regardless of their semantic similarity.

In contrast, word embeddings can capture such semantic similarity as closeness

in their vector space. Furthermore, this resolves the problem of ignoring word order. By using contextualized embeddings, we are able to utilize information about word order. However, their extensions to the representation of a *set* of words are not trivial. Previous studies used approximations to represent a word set. Additive composition [75] is a widely used method that takes the sum of word vectors as the representation of a phrase and a sentence [90]. Although it is very simple and effective, it is unclear how additive composition holds properties as a *word set*. The degree of overlap among word sets has been measured by average cosine similarity [112] and optimal transportation cost [60, 110]. These measures, which are based on element-level approximations, ignore the property of whole sets and such set operations as union and intersection. However, set operations are critical because they can be general-purpose tools in NLP. For example, a word set expansion task called *Text Concept Set Retrieval* [111] and a sentence similarity task called *Semantic Textual Similarity* (STS) [4] respectively require the calculation of the degrees of membership and set similarity.

In this thesis, we propose a novel formulation of set operations in the framework of modern word embeddings. In a pre-trained word embedding space, our proposed method uses linear subspaces to represent sets and set operations, including union, ($\cup$), intersection ($\cap$), complement ($\overline{\mathrm{A}}$), set membership ($\in$), and set similarity (Similarity(A, B)). The proposed method has the following advantage: word order can be taken into account by a set of dynamic embeddings such as BERT. It also has an advantage in downstream tasks; it can be applied to various tasks simply using the set operations defined on the pre-trained embedding space without additional fine-tuning.

The contribution of this work is two-fold: (1) Based on the idea of representing a word set as a subspace, we define general set operations in a pre-trained embedding space and demonstrate that they work as linguistic set operations. (2) Based on the definition of set operations, we propose a soft membership function and set similarity. Experiments on both sentence similarity and concept set retrieval tasks showed that these proposed methods outperformed other methods without additional fine-tuning.

4.2 Required Set Operations

We formulate various set operations in a pre-trained word embedding space in an unsupervised manner. Among many kinds of required operations for practical NLP applications, this work focuses on *set membership*:

$$\begin{aligned} &\text{Color} = \{\textit{red}, \textit{blue}, \textit{green}, \textit{orange}, \dots\}, \\ &\text{Fruit} = \{\textit{apple}, \textit{orange}, \textit{peach}, \dots\}, \\ &\textit{orange} \in \text{Color} \cap \text{Fruit}, \end{aligned} \tag{4.1}$$

and *set similarity*:

$$\begin{aligned} &\text{A} = \{\textit{A}, \textit{boy}, \textit{walks}, \textit{in}, \textit{this}, \textit{park}\}, \\ &\text{B} = \{\textit{The}, \textit{kid}, \textit{runs}, \textit{in}, \textit{the}, \textit{square}\}, \\ &\text{Similarity}(\text{A}, \text{B}). \end{aligned} \tag{4.2}$$

For this purpose, we require representations of the following given a word embedding space*:

An element and a set of elements The representations of an element and a set of elements are the most basic ones. To exploit word embeddings, we represent a word (e.g., *orange*) as an element and a group of words (e.g., $\{\textit{red}, \textit{blue}, \textit{green}, \textit{orange}, \dots\}$) as a word set.

Basic operations on sets We need three basic set operations: *intersection* ($\text{A} \cap \text{B}$), *union* ($\text{A} \cup \text{B}$), and *complement* ($\overline{\text{A}}$). They allow us to represent various sets using such different combinations as $\text{Color} \cap \text{Fruit}$ (Eq. 4.1).

Quantification of set membership *Set membership* denotes a relation in which word w is an element of set A, i.e., $w \in \text{A}$. We quantify it based on vector representations. Although the membership is typically a binary decision identical to that in a symbolic space, it can also be measured by the degree of closeness in a continuous vector space.

*These operations do not include some set operations such as cardinality, but are sufficient for expressing the practical forms of sets such as Equation 4.2.

Set similarity The similarity between two sentences $\text{Similarity}(\text{A}, \text{B})$ can be calculated by set similarity [52] as follows:

$$\text{Johnson}(\text{A}, \text{B}) = \frac{|\text{A} \cap \text{B}|}{|\text{A}|} + \frac{|\text{A} \cap \text{B}|}{|\text{B}|}. \tag{4.3}$$

Here each sentence is treated as a set of its words. We quantify it using the degree of overlap between the two sets using the number of shared elements. The overlap should be considered in a continuous vector space, in contrast to the exact matching of elements in a symbolic space. In the next section, we give a detailed formulation to achieve these set operations in a word embedding space.

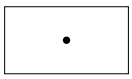
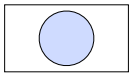
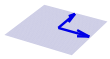
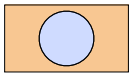
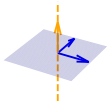
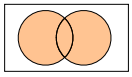
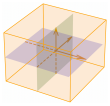
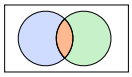
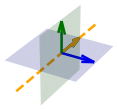
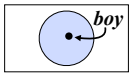
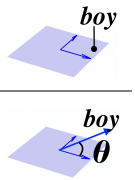
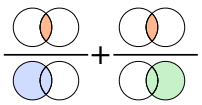
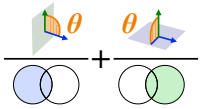
Symbolic Set Representation		Subspace-based Set Representation	
Element *king*		Vector $\boldsymbol{v}_{king}$	
Set $\text{Male} = \{king, man, \dots\}$		Subspace $\mathbb{S}_{\text{Male}} = \text{span}\{\boldsymbol{v}_{king}, \boldsymbol{v}_{man}, \dots\}$	
Complement $\overline{\text{Male}}$		Orthogonal complement $(\mathbb{S}_{\text{Male}})^{\perp}$	
Union $\text{Male} \cup \text{Female}$		Sum space $\mathbb{S}_{\text{Male}} + \mathbb{S}_{\text{Female}}$	
Intersection $\text{Color} \cap \text{Fruit}$		Intersection $\mathbb{S}_{\text{Color}} \cap \mathbb{S}_{\text{Fruit}}$	
Set membership $boy \in \text{Male}$		Hard membership function $\mathbb{1}_{\text{hard}}(\boldsymbol{v}_{boy}, \mathbb{S}_{\text{Male}})$ Soft membership function $\mathbb{1}_{\text{soft}}(\boldsymbol{v}_{boy}, \mathbb{S}_{\text{Male}})$	
Word set similarity $\text{Johnson}(\text{A}, \text{B})$		Word vector set similarity $\text{SubspaceJohnson}(\text{A}, \text{B})$	

Table 4.1: Correspondence between symbolic set representations and subspace-based set representations: We demonstrate that union, intersection, and complement, which are formulated in quantum logic, and our new formulations of set membership and word set similarity hold in pre-trained word embedding space.

4.3 Subspace-based Set Representations

We propose the representations of a word set and set operations based on *quantum logic* [12]. They hold advantages of geometric properties in an embedding space, and the set operations are guaranteed to hold for the laws of a set defined in quantum logic.

4.3.1 Quantum logic

While word embedding represents a word' s meaning as a vector in linear space, quantum mechanics similarly represents a quantum state as a vector in linear space. These two intuitively different fields are very close to each other in terms of the representation and the operation of information.

Quantum logic [12] theory describes quantum mechanical phenomena. Intuitively, it is a framework for *set operations in a vector space*. In quantum logic, a set of vectors is represented as a linear subspace in a Hilbert space, and such set operations as union, intersection, and complement are defined as operations on subspaces. The set operations defined in quantum logic are guaranteed to hold for such laws as De Morgan's laws: $\overline{(\mathrm{A} \cap \mathrm{B})} = \overline{\mathrm{A}} \cup \overline{\mathrm{B}}$ and $\overline{(\mathrm{A} \cup \mathrm{B})} = \overline{\mathrm{A}} \cap \overline{\mathrm{B}}$, idempotent law: $\mathrm{A} \cap \mathrm{A} = \mathrm{A}$, and double complement: $\overline{\overline{\mathrm{A}}} = \mathrm{A}$.

4.3.2 Set Operations in an Embedding Space

The representations of an element, a set, and such set operations as union, intersection, and complement in quantum logic can be applied directly in a word embedding space because it is a Euclidean space and therefore also a Hilbert space. However, since set similarity and set membership for a word embedding space are still missing in quantum logic, we propose a novel formulation of those operations using subspace-based representations, which is consistent with quantum logic. The correspondence between symbolic and subspace-based set operations is shown in Table 4.1.

An element and a set of elements Let $\mathbb{R}^n$ be a n-dimensional word embedding space (Euclidean space), let $\mathrm{A} = \{w_1, w_2, \dots\}$ be a set of words, and let

$\boldsymbol{v}_w \in \mathbb{R}^n$ be a word vector corresponding to w. As discussed in Section 4.2, we first formulate the representation of a word and a word set. In quantum logic, an element is represented by a vector, and a set is represented by a subspace spanned by the vectors corresponding to its elements. Here we assume an element, i.e., word w, is represented by word vector $\boldsymbol{v}_w$, and a word set is represented by linear subspace $\mathbb{S}_\mathrm{A} \subset \mathbb{R}^n$ spanned by word vectors:

$$\mathbb{S}_\mathrm{A} = \mathbb{S}_{\{w_1, w_2, \dots\}} \coloneqq \mathrm{span}\{\boldsymbol{v}_{w_1}, \boldsymbol{v}_{w_2}, \dots\}. \tag{4.4}$$

Hereinafter we simply refer to *linear subspace* as *subspace*. The pseudocode for computing the basis of a subspace is shown in Algorithm 2.

Basic operations on sets The *complement* of set A, denoted by $\overline{\mathrm{A}}$, is represented by the orthogonal complement of subspace $\mathbb{S}_\mathrm{A}$:

$$\mathbb{S}_{\overline{\mathrm{A}}} \coloneqq (\mathbb{S}_\mathrm{A})^\perp = \{\boldsymbol{u} \mid \exists \boldsymbol{v} \in \mathbb{S}_\mathrm{A}, \boldsymbol{u} \cdot \boldsymbol{v} = 0\}. \tag{4.5}$$

The *union* of two sets, A and B, denoted by $\mathrm{A} \cup \mathrm{B}$, is represented by the sum space of two subspaces, $\mathbb{S}_\mathrm{A}$ and $\mathbb{S}_\mathrm{B}$:

$$\mathbb{S}_{\mathrm{A} \cup \mathrm{B}} \coloneqq \mathbb{S}_\mathrm{A} + \mathbb{S}_\mathrm{B} = \{\boldsymbol{u} + \boldsymbol{v} \mid \boldsymbol{u} \in \mathbb{S}_\mathrm{A}, \boldsymbol{v} \in \mathbb{S}_\mathrm{B}\}. \tag{4.6}$$

The *intersection* of two sets, A and B, denoted by $\mathrm{A} \cap \mathrm{B}$, is represented by the intersection of two subspaces, $\mathbb{S}_\mathrm{A}$ and $\mathbb{S}_\mathrm{B}$:

$$\mathbb{S}_{\mathrm{A} \cap \mathrm{B}} \coloneqq \mathbb{S}_\mathrm{A} \cap \mathbb{S}_\mathrm{B} = \{\boldsymbol{v} \mid \boldsymbol{v} \in \mathbb{S}_\mathrm{A}, \boldsymbol{v} \in \mathbb{S}_\mathrm{B}\}. \tag{4.7}$$

The basis of the intersection can be computed based on singular value decomposition (SVD). The bases are the vectors shared by the two input subspaces. The pseudocodes for computing the basis of the complement, the union, and the intersection are shown in Algorithms 3, 5, and 4.

4.3.3 Quantification of Set Membership

The set membership of a word in a word embedding space (e.g., *boy* $\in$ Male) can be represented by the inclusion of a word vector into a subspace (e.g., $\boldsymbol{v}_{boy} \in \mathbb{S}_\mathrm{Male}$)

Algorithm 2 Subspace

Input: Word embeddings to span subspace $\mathbb{S}_{\mathrm{A}}$: $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(k)} \in \mathbb{R}^{1\times d}$
Output: $\mathbf{S}_{\mathrm{A}}$: A matrix with bases on subspace $\mathbb{S}_{\mathrm{A}}$
$\mathbf{A} \in \mathbb{R}^{k\times d} \leftarrow \textsc{stack\_\_rows}(\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(k)})$
$\mathbf{S}_{\mathrm{A}} \in \mathbb{R}^{r\times d} \leftarrow \textsc{ortho\_\_normalization}(\mathbf{A})$ ▷ r is the rank of $\mathbf{A}$

Algorithm 3 Complement

Input: $\mathbf{S}_{\mathrm{A}} \in \mathbb{R}^{k\times d}$: A matrix with bases on subspace $\mathbb{S}_{\mathrm{A}}$
Output: $\mathbf{S}_{\overline{\mathrm{A}}}$: A matrix with bases on orthogonal complement $\mathbb{S}_{\overline{\mathrm{A}}}$
$\mathbf{U} \in \mathbb{R}^{d\times d}, \mathbf{\Sigma} \in \mathbb{R}^{d\times k}, \mathbf{V} \in \mathbb{R}^{k\times k} \leftarrow \mathrm{SVD}({\mathbf{S}_{\mathrm{A}}}^{\top})$
$\mathbf{S}_{\overline{\mathrm{A}}} \leftarrow \varnothing$
for $i = \{1, ..., d\}$ **do** ▷ Stack the bases of the orthogonal complement to $\mathbf{S}_{\overline{\mathrm{A}}}$
  if $i > k$ **then**
    $\mathbf{S}_{\overline{\mathrm{A}}} \leftarrow \textsc{concatenate\_\_rows}(\mathbf{S}_{\overline{\mathrm{A}}}, \boldsymbol{u}^{(i)})$ ▷ $\boldsymbol{u}^{(i)}$ is i-th row of $\mathbf{U}^{\top}$.
  end if
end for

and given by the following indicator function:

$$\mathbb{1}_{\mathrm{hard}}(\boldsymbol{v}_w, \mathbb{S}_{\mathrm{A}}) \coloneqq \begin{cases} 1 & (\boldsymbol{v}_w \in \mathbb{S}_{\mathrm{A}}), \\ 0 & (\boldsymbol{v}_w \notin \mathbb{S}_{\mathrm{A}}). \end{cases} \tag{4.8}$$

However, this binary decision fails to exploit the geometric properties of the word embedding space regarding semantic similarity. Suppose we quantify the degree of membership of word *boy* for word set Male consisting of many masculine nouns other than *boy*. Even if $\boldsymbol{v}_{boy}$ is located very close to $\mathbb{S}_{\mathrm{Male}}$ due to its semantic similarity to masculine nouns, $\mathbb{1}_{\mathrm{hard}}(\boldsymbol{v}_{boy}, \mathbb{S}_{\mathrm{Male}})$ must return 0 because $\boldsymbol{v}_{boy}$ must not be located *exactly* on subspace $\mathbb{S}_{\mathrm{Male}}$ defined by Male. It must return 1 based on the masculine property of word *boy*. Such *hard* membership defined by Eq. 4.8 is incompatible with an embedding space. Instead, we define another membership function called *soft membership* $\mathbb{1}_{\mathrm{soft}}$ that returns continuous values from 0 to 1 depending on the following minimum angle between vector $\boldsymbol{v}_w$ and subspace $\mathbb{S}_{\mathrm{A}}$:

$$\mathbb{1}_{\mathrm{soft}}(\boldsymbol{v}_w, \mathbb{S}_{\mathrm{A}}) \coloneqq \max \left\{ \frac{|\boldsymbol{u} \cdot \boldsymbol{v}_w|}{\|\boldsymbol{u}\| \|\boldsymbol{v}_w\|} \middle| \ \boldsymbol{u} \in \mathbb{S}_{\mathrm{A}} \right\}. \tag{4.9}$$

Algorithm 4 Intersection

Input: $\mathbf{S}_{\mathrm{A}} \in \mathbb{R}^{k \times d}$: A matrix with bases on subspace $\mathbb{S}_{\mathrm{A}}$
Input: $\mathbf{S}_{\mathrm{B}} \in \mathbb{R}^{l \times d}$: A matrix with bases on subspace $\mathbb{S}_{\mathrm{B}}$ $(l \geq k)$
Input: α: Threshold below which the cosine of canonical angles is considered zero.
Output: $\mathbf{S}_{\mathrm{A} \cap \mathrm{B}}$: A matrix with bases on intersection $\mathbb{S}_{\mathrm{A}} \cap \mathbb{S}_{\mathrm{B}}$

```
M ∈ ℝ^{k×l} ← S_A S_B^⊤
U ∈ ℝ^{k×k}, Σ ∈ ℝ^{k×l}, V ∈ ℝ^{l×l} ← SVD(M)
    ▷ Cosine of the canonical angle of the intersection is singular value σ_i of S_A S_B^⊤
W ← ∅
for σ_i ∈ {σ_1, ..., σ_k} do
    if |σ_i − 1| < α then          ▷ Extract the bases for which cosine is almost 1.0
        W ← CONCATENATE_ROWS(W, u^(i))          ▷ u^(i) is i-th row of U^⊤
    end if
end for
S_{A∩B} ∈ ℝ^{m×d} ← (S_A^⊤ W^⊤)^⊤          ▷ m is number of rows in W ∈ ℝ^{m×d}
```

Algorithm 5 Union

Input: $\mathbf{S}_{\mathrm{A}} \in \mathbb{R}^{k \times d}$: A matrix with bases on subspace $\mathbb{S}_{\mathrm{A}}$
Input: $\mathbf{S}_{\mathrm{B}} \in \mathbb{R}^{l \times d}$: A matrix with bases on subspace $\mathbb{S}_{\mathrm{B}}$
Output: $\mathbf{S}_{\mathrm{A} \cup \mathrm{B}}$: A matrix with bases on union $\mathbb{S}_{\mathrm{A}} \cup \mathbb{S}_{\mathrm{B}}$

```
M ∈ ℝ^{(k+l)×d} ← CONCATENATE_ROWS(S_A, S_B)
S_{A∪B} ∈ ℝ^{r×d} ← ORTHO_NORMALIZATION(M)          ▷ r is rank of M
```

This captures the degree of membership between a word and a word set, represented by the angle between a word vector and a subspace. It is a natural extension of hard membership $\mathbb{1}_{\mathrm{hard}}$, i.e., $\mathbb{1}_{\mathrm{soft}}$ returns 1 when $\boldsymbol{v}_w \in \mathbb{S}_{\mathrm{A}}$ and 0 when $\boldsymbol{v}_w$ is orthogonal to $\mathbb{S}_{\mathrm{A}}$.

Soft membership uses rich information from pre-trained word embedding space. Note that soft membership does not compute cosine similarity with each word in a given word set but rather is a similarity based on the angle with all the vectors in the subspace spanned by the given word vectors.

Soft membership, i.e., the cosine of the first canonical angle can be computed by SVD [59]. The pseudocode for computing soft membership is shown in Algorithm 6.

4.3.4 Set Similarity

Suppose we quantify the set similarity between A = {*A*, *boy*, *walks*, *in*, *this*, *park*} and B = {*The*, *kid*, *runs*, *in*, *the*, *square*}, which represent semantically similar sentences. Although we can use such symbolic set similarities as Johnson(A, B) (Eq. 4.3), unfortunately, such symbolic set operations do not work for this example. These semantically similar sentences share only one word {*in*} between A and B: $|\mathrm{A} \cap \mathrm{B}| = 1$. To address this problem, we propose a new *soft* set similarity measure, named SubspaceJohnson, as a natural extension of Johnson similarity by utilizing our proposed subspace-based methods.

Make membership functions soft First, note that each term of the Johnson similarity (Eq. 4.3) can be rewritten using the hard membership function (Eq. 4.8):

$$\frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A}|} = \frac{\sum_{a \in \mathrm{A}} \mathbb{1}_{\mathrm{hard}}(\boldsymbol{v}_a, \mathbb{S}_\mathrm{B})}{|\mathrm{A}|} \tag{4.10}$$

This transformation is from the definition of intersection: $\mathrm{A} \cap \mathrm{B} \coloneqq \{\mathrm{a} \in \mathrm{A} \mid \mathrm{a} \in \mathrm{B}\}$. Here the soft membership function introduced in the previous section (Eq. 4.9) can be used to naturally replace the hard membership function. That is, the Johnson similarity can be *softened*:

$$\frac{\sum_{a \in \mathrm{A}} \mathbb{1}_{\mathrm{soft}}(\boldsymbol{v}_a, \mathbb{S}_\mathrm{B})}{|\mathrm{A}|} + \frac{\sum_{b \in \mathrm{B}} \mathbb{1}_{\mathrm{soft}}(\boldsymbol{v}_b, \mathbb{S}_\mathrm{A})}{|\mathrm{B}|}. \tag{4.11}$$

The numerator of the Johnson similarity is the cardinality of symbolic intersection $|\mathrm{A} \cap \mathrm{B}|$.In quantum logic, cardinality is the dimension of a subspace, and so the direct way is to use the dimension of intersection $\dim(\mathbb{S}_\mathrm{A} \cap \mathbb{S}_\mathrm{B}) = \mathrm{rank}(\mathbf{S}_{\mathrm{A} \cap \mathrm{B}})$ as the numerator. We tried to use the dimension of the intersection as the numerator, but since the dimension takes discrete values, the range of the set similarity is quite limited. We tackled this problem using the sum of soft membership $\mathbb{1}_{\mathrm{soft}}$ which derives continuous values instead of ranks.

Consider word weights Each term of the Johnson similarity (Eq. 4.3) can be extended by introducing the following weights:

$$\frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A}|} = \frac{\sum_{\mathrm{a} \in \mathrm{A}} \mathrm{weight}(a) \mathbb{1}_{\mathrm{hard}}(\boldsymbol{v}_a, \mathbb{S}_\mathrm{B})}{\sum_{a \in \mathrm{A}} \mathrm{weight(a)}}, \tag{4.12}$$

Algorithm 6 Soft membership function $\mathbb{1}_{\text{soft}}$

Input: $\mathbf{S}_{\text{A}} \in \mathbb{R}^{k \times d}$: A matrix with bases on subspace $\mathbb{S}_{\text{A}}$
Input: Word embedding $\boldsymbol{v}_w \in \mathbb{R}^{1 \times d}$
Output: Degree of membership m

$\boldsymbol{v}_w \in \mathbb{R}^{1 \times d} \leftarrow \frac{\boldsymbol{v}_w}{||\boldsymbol{v}_w||}$
$\mathbf{U} \in \mathbb{R}^{k \times k}, \boldsymbol{\Sigma}^{k \times 1} \in \mathbb{R}, \mathbf{V} \in \mathbb{R} \leftarrow \text{SVD}(\mathbf{S}_{\text{A}} {\boldsymbol{v}_w}^{\top})$
$m \leftarrow \text{MAX}(\boldsymbol{\Sigma})$

where weight(a) $\equiv 1$. This means that each word a $\in$ A is implicitly assigned an equal weight of 1. Instead, we can use the L2 norm for this weight, i.e., weight(a) $\equiv ||\boldsymbol{v}_{\text{a}}||_2$, inspired by the findings that the norm of a word vector implicitly encodes the word importance [110]. By using norms as word weights, we can exploit the geometric properties of word vectors without omission. That is, the orientation of the word vector was used for subspace configuration (Eq. 4.4) and membership calculation (Eq. 4.9), and its length was used for the word weights.

Summary The following is our proposed measure, named **SubspaceJohnson**[†]:

$$\begin{aligned} \text{SubspaceJohnson}(\text{A}, \text{B}) \coloneqq & \frac{\sum_{a \in \text{A}} ||\boldsymbol{v}_a||_2 \ \mathbb{1}_{\text{soft}}(\boldsymbol{v}_a, \mathbb{S}_{\text{B}})}{\sum_{a \in \text{A}} ||\boldsymbol{v}_a||_2} \\ & + \frac{\sum_{b \in \text{B}} ||\boldsymbol{v}_b||_2 \ \mathbb{1}_{\text{soft}}(\boldsymbol{v}_b, \mathbb{S}_{\text{A}})}{\sum_{b \in \text{B}} ||\boldsymbol{v}_b||_2}. \end{aligned} \tag{4.13}$$

The pseudocode for computing SubspaceJohnson is shown in Algorithm 7.

[†]In addition to Johnson similarity, there are other measures of set similarities such as Jaccard, Dice, and Simpson's Coefficient. We extended these to set similarity using subspaces, similar to Johnson similarity, and experimented with them on the STS task, but they did not perform as well as Johnson in terms of performance.

Algorithm 7 SubspaceJohnson

Input: Word embeddings for first sentence $\boldsymbol{a}^{(1)}, \boldsymbol{a}^{(2)}, \ldots, \boldsymbol{a}^{(k)} \in \mathbb{R}^{1 \times d}$

Input: Word embeddings for second sentence $\boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}, \ldots, \boldsymbol{b}^{(l)} \in \mathbb{R}^{1 \times d}$

Output: Similarity score s

$\mathbf{S}_{\mathrm{A}} \leftarrow \textsc{subspace}(\boldsymbol{a}^{(1)}, \boldsymbol{a}^{(2)}, \ldots, \boldsymbol{a}^{(k)})$

$\mathbf{S}_{\mathrm{B}} \leftarrow \textsc{subspace}(\boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}, \ldots, \boldsymbol{b}^{(l)})$

$x \leftarrow \sum_{i=1}^{k}(||\boldsymbol{a}^{(i)}||_2 \mathbb{1}_{\mathrm{soft}}(\mathbf{S}_{\mathrm{A}}, \boldsymbol{a}^{(i)})) \;/\; \sum_{i=1}^{k} ||\boldsymbol{a}^{(i)}||_2$

$y \leftarrow \sum_{j=1}^{l}(||\boldsymbol{b}^{(j)}||_2 \mathbb{1}_{\mathrm{soft}}(\mathbf{S}_{\mathrm{B}}, \boldsymbol{b}^{(j)})) \;/\; \sum_{j=1}^{l} ||\boldsymbol{b}^{(j)}||_2$

$s \leftarrow x + y$

4.4 Experiments

We conducted the following experiments to evaluate our proposed method from the following three perspectives: (1) whether the subspace-based formulation of sets and set operations behaves appropriately as a tool for computing the meaning of the language (Section 4.4.1); (2) whether our soft membership function is empirically helpful (Section 4.4.2); and (3) whether our soft set similarity function is empirically helpful (Section 4.4.3).

Word embeddings We used the most standard pre-trained word embeddings in all of our experiments: 300-dimensional **GloVe**[‡] [84], which was pre-trained with Common Crawl (840B tokens), and 300-dimensional **word2vec**[§] [75], which was pre-trained with Google News.

In Semantic Textual Similarity tasks, we also used 768-dimensional $\textbf{BERT}_{\text{base}}$[¶] [30], which was pre-trained with BookCorpus and Wikipedia, and **SimCSE** [35]. SimCSE is a very powerful contrastive learning framework in recent studies. It applies contrastive learning to a pre-trained language model as its fine-tuning to obtain sentence embeddings. It has an unsupervised model using Wikipedia as unlabeled text and a supervised variant using MNLI and SNLI as labeled datasets. We used $\textbf{SimCSE-BERT}_{\text{base}}$[‖], which was fine-tuned using $[CLS]$ representation as sentence embedding. In both $\text{BERT}_{\text{base}}$ and $\text{SimCSE-BERT}_{\text{base}}$, we used embeddings in the last layer.

4.4.1 Qualitative Demonstration of Subspace-based Set Operations

Next, we describe the appropriateness of our proposed subspace-based set representation from the viewpoint of NLP. For example, does subset $\mathbb{S}_{\text{Male}}$ formed by word set $\text{Male} = \{king, man, \dots\}$ well represent the abstract notion of masculine? If the proposed method works properly, word vector $\boldsymbol{v}_{boy}$ that shares the concept

[‡] https://nlp.stanford.edu/projects/glove/
[§] https://code.google.com/archive/p/word2vec/
[¶] https://huggingface.co/bert-base-uncased
[‖] https://huggingface.co/princeton-nlp/unsup-simcse-bert-base-uncased

of masculine should be located around $\mathbb{S}_{\text{Male}}$ *even if word boy is not in the given set* Male. In this section, we qualitatively tested whether the proposed method adequately represents the linguistic characteristics of a word set.

Experimental procedure The experiment was conducted as follows, explained by an example from set Male.

1. Assume the following given word set:
   $\text{Male} = \{king, boy, man, son \dots\}$

2. Randomly divides it into two subsets: an input subset for generating a subspace and a test subset for validation.
   $\text{Male}^{\text{input}} = \{king, man, \dots\}$
   $\text{Male}^{\text{test}} = \{boy, son, \dots\}$

3. Span subspace (e.g., $\mathbb{S}_{\text{Male}}$) with the input subset (e.g., $\text{Male}^{\text{input}}$).

4. Find word vectors close to the subspace. That is, find $\boldsymbol{v}_w$ such that $\mathbb{1}_{\text{soft}}(\boldsymbol{v}_w, \mathbb{S}_{\text{Male}})$ is large. To this end, we employed a sampling-based procedure. (1) Randomly sample a vector** $\boldsymbol{v}$ from $\mathbb{S}_{\text{Male}}$. (2) Find word w corresponding to word vector $\boldsymbol{v}_w$ that gives largest cosine similarity $\cos(\boldsymbol{v}, \boldsymbol{v}_w)$ for $\boldsymbol{v} \in \mathbb{S}_{\text{Male}}$. (3) Repeat (1) and (2) 100 times.

5. Check whether searched w shares the concept of the set (e.g., Male).

Baseline One might think that just the nearest neighbor search is the best strategy to find a word vector that is close to a given word vector set. As a baseline for the experiment, we used the distance to the closest word vector in a given set of word vectors (**vector-based**). For the complement $\overline{\text{A}}$, we chose bottom-ranked word vectors instead.

As a metric of nearest neighbor search, we used the cosine distance. Cosine is empirically better than Euclidean for computing lexical similarity [110]. Furthermore, the setting of the baseline method became consistent with that of the proposed method, thus maintaining the fairness of the validation.

** A random vector is given by the linear combination of basis $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3 \dots\}$ of the subspace using random coefficients $\{c_1, c_2, c_3 \dots\}$, which were generated from the standard normal distribution.

Table 4.2: Statistics of word sets used in the experiment in Section 4.4.1 of the number of words: A word set was divided into two subsets: words to generate subspace (Input) and the rest for testing (Test).

Concept Label	#Words		
	Input	Test	Total
Male	49	49	98
Female	44	44	88
Fruit	10	11	21
Color	10	10	20

Dataset As datasets, four sets of words were first prepared using synset in WordNet [77]: Male = $\{king, man, \dots\}$, Female = $\{queen, woman, \dots\}$, Color = $\{red, blue, \dots\}$, and Fruit = $\{apple, peach, \dots\}$. See Table 4.2 for the statistics. To create a set that contains as many words as possible with a certain shared concept, some were added manually so that the number of words in a set was at least 20. Next, we created the following four comprehensive sets for the experiment: Male, $\overline{\text{Male}}$, Male $\cup$ Female, and Color $\cap$ Fruit.

Results Table 4.3 shows the top-ranked words found by the proposed and baseline methods. Words labeled with ✓are positive results, and those labeled with × are negative results. This result shows that **our proposed method effectively represents the linguistic characteristics of each given word set.** For example, with the subspace-based method, only nouns with masculine meanings were appropriately found near $\mathbb{S}_{\text{Male}}$. In contrast, with the vector-based method, nouns without masculine meanings (e.g., *fiancee* and *aunts*) were found. As another example, the word vector for *orange* was very close to subspace $\mathbb{S}_{\text{Color}} \cap \mathbb{S}_{\text{Fruit}}$; *orange* obviously has a property of both color and fruit. Note that Color and Fruit were used to span the subspace and do not include *orange* (see Experimental procedure).

The result also suggests that our method is helpful for expanding a word set (e.g., synset) by retrieving words that share its concept. Indeed, the found words *brother-in-law* and *step-father* shared the concept of Male and were close to sub-

Table 4.3: Three nearest neighbors for different word sets: Note that the table only lists words that were not used to span subspaces. ✓✓: words for test set; ✓: words not included in test set but natural; ×: unnatural words; no mark: word that cannot be determined.

	Set	Male	$\overline{\text{Male}}$	Male ∪ Female	Color ∩ Fruit
	Subspace	$\mathbb{S}_{\text{Male}}$	$(\mathbb{S}_{\text{Male}})^{\perp}$	$\mathbb{S}_{\text{Male}} + \mathbb{S}_{\text{Female}}$	$\mathbb{S}_{\text{Color}} \cap \mathbb{S}_{\text{Fruit}}$
Subspace-based	**GloVe**	brother-in-law✓ nephews✓✓ step-father✓	drunken✓ erect✓ nominal✓	moms✓✓ uncle✓✓ granddads✓	orange✓✓ peach✓ kumquat
	word2vec	father✓✓ grandkids son✓✓	Ahnlund✓ Confirm✓ Sakowicz✓	siblings✓ bachelors✓✓ giantess✓	pear_apple× pear✓ orange✓✓
Vector-based	**GloVe**	fiancé✓ fiancee× aunts×	DTDigitized✓ d/web6✓ CamelKarma✓	nephews✓✓ males✓✓ grandchildren✓	magenta× black× colors×
	word2vec	father✓✓ females× wife×	GENERAL_RULES✓ M.Martin_###-###✓ THE_MORNING_MEMO✓	grandmother✓✓ male✓✓ stallion✓✓	black× surged× climbed×

space $\mathbb{S}_{\text{Male}}$, even though they were not included in the input or the test subsets.

4.4.2 Text Concept Set Retrieval Task

Second, we conducted a *quantitative* evaluation of the ability of the proposed method to capture the linguistic features of word sets. We employed an existing set expansion task, the Concept Set Retrieval task [111], to determine whether the soft membership function can retrieve words that should be included in a given word set. In this task, a set of words is given, and the model retrieves words that belong to the set from the vocabulary.

Evaluation procedure Taking a set of masculine nouns as an example, we describe the evaluation procedure by following [111].

1. Assume two sets of words (input and test), which share a certain topic:
   $\mathrm{Set}^{\mathrm{input}} = \{king, man, \dots\}$
   $\mathrm{Set}^{\mathrm{test}} = \{boy, son, \dots\}$

2. Generate a subspace representation with the input word set: $\mathbb{S} \coloneqq \mathbb{S}_{\{king,man,\dots\}}$.

3. Rank all in-vocabulary words $\{w\}$ by their soft membership scores $\mathbb{1}_{\mathrm{soft}}(\boldsymbol{v}_w, \mathbb{S})$.

4. Report several evaluation scores that indicate whether the words in the test set can be ranked higher. We used **Recall@K**, Mean Reciprocal Rank (**MRR**), and **Median** for the metrics.

Baselines We compared several baselines, including methods that require/don' t require training on word sets, to the proposed method.

Random just selects words randomly from the dataset' s vocabulary.

The following methods do not require training on word sets. Note that our subspace-based representation is also such a method. First, **Bayesian Sets** (Bayes Set) [37] is a symbol-based unsupervised method that computes a probabilistic score based on the pointwise mutual information between $w \in$ V and set A. Second, a simple unsupervised baseline with word embeddings uses the nearest neighbors in the embedding space (**w2v Near**). Third, we also compare a method based on fuzzy sets (**Fuzzy Set** by [115]) with the proposed method. Similar to our method, their method is designed to exploit both the flexibility of

Table 4.4: Examples from original dataset (denoted as $\mathbf{D}^{\text{Set}}$) and additional $\mathbf{D}^{\text{Union}}$ and $\mathbf{D}^{\text{Intersection}}$ sets

$\mathbf{D}^{\text{Set}}$	$\mathbf{D}^{\text{Union}}$			$\mathbf{D}^{\text{Intersection}}$		
Set_3	Set_{12}	Set_{51}	$\text{Set}_{12} \cup \text{Set}_{51}$	Set_9	Set_{72}	$\text{Set}_9 \cap \text{Set}_{72}$
daily	motorcycle	island	races	tour	poker	money
newspaper	bike	islands	mainland	open	casino	won
paper	bicycle	mainland	nearby	golf	gambling	players
news	rider	coast	area	championship	lottery	tournament
press	riders	sea	cycling	player	bet	professional
...	...	...	...	...	...	...

Table 4.5: Number of word sets treated in the experiment $\mathbf{D}^{\text{Set}}$ is the original dataset. $\mathbf{D}^{\text{Union}}$ and $\mathbf{D}^{\text{Intersection}}$ dataset are newly created.

Set type		(for models that require training on word sets)	
	Test	Train	Dev
$\mathbf{D}^{\text{Set}}$	100	800	100
$\mathbf{D}^{\text{Union}}$	100	-	-
$\mathbf{D}^{\text{Intersection}}$	100	-	-

word vectors and rich set operations. Fuzzy Set represents word set A by max-pooled word vectors $\boldsymbol{s} = \max_{w \in \text{A}} \boldsymbol{v}_w$. Although the Text Concept Set Retrieval task requires computing the degree of a word' s membership for a word set, their method does not provide it. We instead used cosine similarity $\cos(\boldsymbol{v}_w, \boldsymbol{s})$ between word vector $\boldsymbol{v}_w$ of word $w \in \text{V}$ and $\boldsymbol{s}$ as the degree of membership to apply fuzzy sets to the task.

Other baselines that use the vector representations of word sets are trained using the following word set dataset. **DeepSets** [111] is a popular model for the representation of a set. The recently proposed **LAF** [83] is based on more complex mappings than such aggregation functions as sum and average, which DeepSets used. We also included **LSTM** [45] and models with architectures similar to DeepSets (**NN-max, NN-sum-con**, and **NN-man-con**) [111].

Dataset We used a previously created dataset [111], which was denoted by "LDA-1k, Vocab = 17k.". This dataset ($\mathbf{D}^{\text{Set}}$) contains 100 word sets, each of which consists of 50 words sampled from a common topic[††]. Five pre-determined words from each set were used as an input set S^{input}. An additional 800 word sets were used to train the models that require training on word sets. Table 4.4 shows an example of the data, and Table 4.5 shows the statistics of the datasets.

To evaluate the union and intersection sets, we prepared additional data through the union and intersection operations on two randomly-selected word sets from the original word sets ($\mathbf{D}^{\text{Set}}$). The number of words in each set in $\mathbf{D}^{\text{Union}}$ was limited to 50 to match the original dataset ($\mathbf{D}^{\text{Set}}$). The number of words in each set in $\mathbf{D}^{\text{Intersection}}$ was set to a minimum of 10. In experiments on union and intersection, we compared the proposed method only with Fuzzy Set. The proposed method and Fuzzy Set can induce representations for the union and intersection using set operations defined in the word embedding space; the others cannot do so directly. See Tables 4.4 and 4.5 for examples and statistics of the datasets.

[††]This work used Latent Dirichlet Allocation [13] (LDA) as a topic model.

Results Table 4.6 shows the experimental results on original dataset ($\mathbf{D}^{\text{Set}}$). Here **our subspace-based method (SoftMember) was the best among the methods that did not require training on word sets.** Moreover, surprisingly, **SoftMember equaled or outperformed the state-of-the-art methods that were trained using the word set dataset**, such as DeepSets and LAF. The results suggest that combining off-the-shelf pre-trained word vectors with appropriate set-oriented operations makes linguistic computation on word sets feasible without additional costly training.

Table 4.7 gives the results of the $\mathbf{D}^{\text{Union}}$ and $\mathbf{D}^{\text{Intersection}}$ datasets. **The proposed method outperformed Fuzzy Set in most metrics.** As methods for achieving set operations in vector spaces, the proposed subspace-based method is empirically more promising than the existing fuzzy set-based method.

Table 4.6: Results of text concept set retrieval task: All baseline results were taken from [111] (♠) and [83] (♢).

		$\mathbf{D}^{\text{Set}}$			
		Recall		**MRR**	**Med.**
	Train with word sets	**@100**	**@1k**		
Random♠		0.6	5.9	0.001	8520
NN-max♠	✓	22.5	53.1	0.023	779
NN-sum-con♠	✓	19.8	48.5	0.021	1110
NN-max-con♠	✓	16.9	46.6	0.018	1250
DeepSets♠	✓	24.2	54.3	0.025	696
LSTM♢	✓	21.5	52.6	0.022	690
LAF♢	✓	26.6	54.5	0.030	650
Symbol × Set Representation					
Bayes Set♠		11.9	37.2	0.007	2848
Embedding (word2vec)					
w2v Near♠		28.1	54.7	**0.021**	641
Embedding × Set Representation					
word2vec	Fuzzy Set	19.9	47.2	0.017	1240
	SoftMember	29.7	58.9	0.017	478
GloVe	Fuzzy Set	30.9	69.0	0.023	320
	SoftMember	**35.7**	**72.7**	0.020	**246**

Table 4.7: Results of concept set retrieval task on union and intersection

		$\mathbf{D}^{\text{Union}}$				$\mathbf{D}^{\text{Intersection}}$			
		Recall		**MRR**	**Med.**	**Recall**		**MRR**	**Med.**
		@100	**@1k**			**@100**	**@1k**		
Random		0.6	6.0	0.001	8422	0.2	6.6	0.000	7929
w2v Near		17.5	34.3	**0.015**	3270	23.5	40.8	0.018	3304
Embedding × Set Representation									
word2vec	Fuzzy Set	2.8	17.1	0.003	4426	4.7	20.9	0.002	3420
	SoftMember	18.4	46.9	0.004	1202	25.7	45.7	0.017	1445
GloVe	Fuzzy Set	5.4	32.0	**0.005**	2347	32.5	75.0	0.023	255
	SoftMember	**24.4**	**68.3**	**0.005**	**407**	**44.2**	**83.7**	**0.028**	**149**

4.4.3 Semantic Textual Similarity Task

Can our proposed method capture the similarity between two word sets? In this section, we examine the effectiveness of the proposed set similarity (Section 4.3.4) through a Semantic Textual Similarity (STS) task.

Experimental procedure An STS task calculates the similarity between two sentences. Its evaluation is based on the correlation between the similarity calculated by the model and corresponding human judgments. We used datasets from the SemEval shared task 2012-2016 [1–5], STS benchmark (STS-B) [21], and SICK-Relatedness (SICK-R) [73]. The correlation was measured by Pearson's r and Spearman's ρ coefficients[‡‡].

Baselines We compared our method **SubspaceJohnson** (**Subspace**) (Eq. 4.13) with the baselines, which included unsupervised and supervised methods. The unsupervised baselines included **Avg-cos** [8], the cosine similarity between the averaged vectors, **CLS-cos** [35], the cosine similarity between the $[CLS]$ representations of the pre-trained language model, **DynaMax** [115], a set similarity based on fuzzy sets, **Word Mover's Distance (WMD)** [60], an alignment-based similarity based on optimal transport cost, and **Word Rotator's Distance (WRD)** [110], an alignment-based similarity that improves WMD. The following are similarity measures using pre-trained word embeddings.

Johnson similarity [52] and **Jaccard index** [48] are also set similarities that regard words as symbols.

IS-BERT [113] is another unsupervised baseline. **InferSent** [28], **Universal Sentence Encoder (USE)** [20], and **Sentence-BERT (SBERT)** [90] are supervised baselines. **Contrastive Tension (CT)** [19] and **SimCSE** are very recently proposed powerful methods that are included in both unsupervised and supervised baselines.

[‡‡]Pearson's r tends to be used for methods using static embeddings, and ρ tends to be used for recent methods using dynamic embeddings. In this work, we used both r and ρ to compare previous studies using static and dynamic embeddings.

Main results The results are shown in Table 4.8. **The proposed method worked the best among the similarity metrics** and demonstrated higher overall correlations than a simple baseline of Avg-cos. It also outperformed the strong baseline of DynaMax with GloVe, word2vec, BERT, and SimCSE-BERT. This result suggests that the proposed subspace-based approach represents a set and set operations better than the fuzzy set-based approach in embedding space. The advantage of the proposed method over symbolic set similarity (Johnson similarity) shows that subspace-based set operations in an embedding space effectively extend the set representation.

The proposed method (SubspaceJohnson) achieved high performance without additional fine-tuning.

In particular, the SubspaceJohnson and SimCSE-BERT combination was very powerful and outperformed such supervised methods as InferSent, Universal Sentence Encoder, and SentenceBERT.

Sentence vector vs. Word vectors Even though recent trends in sentence similarity have focused on creating sentence vectors, our results suggest that dealing directly with vector sets is superior to creating sentence vectors. For example, focusing on the use of SimCSE-BERT embeddings, the method that calculated set similarity using token embeddings (as in SubspaceJohnson and DynaMaxJaccard) outperformed the cosine similarity between sentence vectors (CLS-cos) used in [35].

Symbolic set similarity vs. Embedding-based set similarity Next, we address the performance effectiveness of symbolic set similarity without embeddings. Both the Jaccard coefficient and Johnson similarity scores are less than 55, where the latter score is slightly better. Our proposed method, SubspaceJohnson, which extends Johnson similarity using embeddings, achieved a maximum score of 76.61. This result depends on pre-trained embedding, and with good embedding such as SimCSE-BERT, the proposed method is very effective. Using other good embeddings will likely increase the score further.

Ablation studies We also investigated the extent to which each different component of the proposed method affected performance. Table 4.9 shows ablation

studies for SubspaceJohnson. The results identify a strong weight contribution (Eq. (4.12)) to the performance of SubspaceJohnson.

Table 4.8: Correlation coefficient for STS task: Pearson's r and Spearman's ρ are listed as $r \times 100 / \rho \times 100$. Unreported values were marked with "-". Models with identical pre-trained embedding with the highest values are shown in bold. Baseline scores were taken from [115] (♠), [110] (‡), [35] (♢), [90] (♣), [113] (♡). All other experimental results are our results using codes from [35,110,115]. For the STS evaluation protocol, we basically follow [35], except we follow [115] for the GloVe and word2vec results.

Vectors	Sentence	Similarity	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Embedding (unsupervised) × Similarity										
GloVe	Sent vec.	Avg-cos♠	52.1 / -	49.6 / -	54.6 / -	56.1 / -	51.4 / -	- / -	- / -	52.76 / -
	Word vecs.	DynaMax♠	58.2 / -	53.9 / -	65.1 / -	**70.9** / -	**71.1** / -	- / -	- / -	63.83 / -
	Word vecs.	WMD‡	55.74 / -	44.18 / -	60.24 / -	67.11 / -	- / -	52.19 / -	61.91 / -	- / -
	Word vecs.	WRD‡	58.28 / -	48.79 / -	62.31 / -	68.80 / -	- / -	54.03 / -	63.84 / -	- / -
	Word vecs.	**Subspace**	**58.42** / 57.67	**58.30** / 58.96	**65.55** / 64.44	**70.90** / 70.78	68.14 / 68.95	- / -	- / -	**64.26** / 64.16
word2vec	Sent vec.	Avg-cos♠	51.6 / -	58.2 / -	65.6 / -	67.5 / -	64.7 / -	- / -	- / -	61.52 / -
	Word vecs.	DynaMax♠	53.7 / -	59.5 / -	68.0 / -	**74.2** / -	**71.3** / -	- / -	- / -	65.35 / -
	Word vecs.	WMD‡	55.89 / -	44.52 / -	60.24 / -	66.46 / -	- / -	56.10 / -	64.05 / -	- / -
	Word vecs.	WRD‡	**59.14** / -	51.41 / -	65.36 / -	72.39 / -	- / -	72.39 / -	66.31 / -	- / -
	Word vecs.	**Subspace**	54.13 / 56.50	**61.73** / 61.61	**68.61** / 66.68	73.25 / 72.94	69.34 / 70.31	- / -	- / -	**65.41** / 65.61
BERT	Setn vec.	CLS-cos	13.73 / 21.54	31.75 / 32.11	21.56 / 21.28	34.84 / 37.89	40.15 / 44.24	17.18 / 20.29	38.34 / 42.42	28.22 / 31.40
	Setn vec.	Avg-cos	27.09 / 30.87	**58.88** / **59.89**	**50.71** / 47.73	59.39 / 60.29	62.52 / 63.73	47.91 / 47.29	**61.21** / **58.22**	52.53 / 52.57
	Word vecs.	DynaMax	27.97 / 32.25	50.83 / 51.84	44.85 / 43.26	61.41 / 61.60	64.32 / 63.91	46.30 / 45.24	57.02 / 56.00	50.39 / 50.59
	Word vecs.	WMD	23.38 / 23.79	43.61 / 44.34	39.88 / 38.91	52.07 / 53.08	53.82 / 53.17	38.41 / 38.41	49.58 / 50.91	42.96 / 43.23
	Word vecs.	WRD	20.57 / 24.13	49.34 / 50.19	42.57 / 40.96	57.00 / 57.33	57.67 / 57.25	42.70 / 42.11	53.09 / 52.74	46.13 / 46.39
	Word vecs.	**Subspace**	**31.61** / **34.19**	56.98 / 56.88	50.68 / **48.08**	**61.75** / **62.19**	**65.39** / **65.42**	**49.71** / **48.60**	58.17 / 56.80	**53.47** / **53.17**
SimCSE-BERT	Sent vec.	CLS-cos♢	75.69 / **68.40**	82.10 / 82.41	78.10 / 74.38	80.22 / 80.91	77.57 / **78.56**	77.88 / 76.85	**79.86** / **72.23**	78.77 / 76.25
	Sent vec.	Avg-cos	**75.90** / 67.49	82.06 / 82.50	78.43 / 74.09	81.49 / 82.29	76.33 / 77.68	79.30 / 78.63	79.12 / 70.38	78.95 / 76.15
	Word vecs.	DynaMax	75.02 / 66.74	81.63 / 82.07	78.67 / 74.59	**81.83** / 82.36	77.58 / 78.46	80.65 / 79.54	79.35 / 70.86	79.25 / 76.37
	Word vecs.	WMD	70.01 / 65.59	79.27 / 80.18	75.25 / 73.12	78.90 / 79.57	75.96 / 76.25	78.71 / 77.86	74.88 / 70.04	76.14 / 74.66
	Word vecs.	WRD	72.41 / 64.76	80.74 / 80.99	77.43 / 73.42	80.67 / 80.95	76.24 / 76.60	79.70 / 78.34	78.23 / 70.22	77.92 / 75.04
	Word vecs.	**Subspace**	75.35 / 66.07	**82.49** / **82.96**	**79.17** / **74.90**	81.61 / **82.31**	**77.64** / 78.54	**81.25** / **80.13**	79.78 / 71.38	**79.61** / **76.61**
Symbol × Set similarity										
Johnson			40.78 / 43.06	50.14 / 49.84	52.43 / 52.97	68.07 / 67.84	57.95 / 57.59	52.33 / 52.55	57.58 / 54.26	54.18 / 54.02
Jaccard			40.87 / 43.02	49.71 / 48.89	50.37 / 51.18	67.00 / 67.47	58.51 / 57.48	53.10 / 52.60	55.11 / 54.09	53.52 / 53.53
Other sentence vectors (unsupervised)										
IS-BERT$^{\heartsuit}_{\text{base}}$			- / 56.77	- / 69.24	- / 61.21	- / 75.23	- / 70.16	- / 69.21	- / 64.25	- / 66.58
CT-BERT$^{\diamondsuit}_{\text{base}}$			- / 61.63	- / 76.80	- / 68.47	- / 77.50	- / 76.48	- / 74.31	- / 69.19	- / 72.05
Supervised models										
InferSent-GloVe♣			- /52.86	- /66.75	- /62.15	- /72.77	- /66.87	- /68.03	- /65.65	- /65.01
USE♣			- /64.49	- /67.80	- /64.61	- /76.83	- /73.18	- /74.92	- /76.69	- /71.22
SBERT$^{\clubsuit}_{\text{base}}$			- /70.97	- /76.53	- /73.19	- /79.09	- /74.30	- /77.03	- /72.91	- /74.89
SBERT$_{\text{base}}$-flow$^{\diamondsuit}$			- /69.78	- /77.27	- /74.35	- /82.01	- /77.46	- /79.12	- /76.21	- /76.60
SBERT$_{\text{base}}$-whitening$^{\diamondsuit}$			- /69.65	- /77.57	- /74.66	- /82.27	- /78.39	- /79.52	- /76.91	- /77.00
CT-SBERT$^{\diamondsuit}_{\text{base}}$			- /74.84	- /83.20	- /78.07	- /83.84	- /77.93	- /81.46	- /76.42	- /79.39
SimCSE-BERT$^{\diamondsuit}_{\text{base}}$			- /75.30	- /84.67	- /80.19	- /85.40	- /80.82	- /84.25	- /80.39	- /81.57

Table 4.9: Ablation studies on the presence or absence of weights for SubspaceJohnson: Values are Spearman's $\rho \times 100$. Embeddings in this experiment were pre-trained BERT-base and unsupervised SimCSE-BERT. $\diamondsuit$: results from [35].

Vectors	**Similarity**	**STS-B dev.**
BERT	Subspace	
	w/ weight (default)	**57.0**
	w/o weight	56.2
SimCSE-BERT	Subspace	
	w/ weight (default)	**84.5**
	w/o weight	83.8

4.5 Related Work

In this section, we review previous studies that represent a set and set operations in an embedding space.

4.5.1 Set Representation

Approach using symbols Symbol-based similarity between word sets has been used, such as Jaccard coefficient [48, 72, 96] and TF-IDF-based cosine similarity [53]. Bayesian Sets [37] give a membership score as the probability of the set membership of a word set using pointwise mutual information. Unfortunately, symbol-based methods cannot capture the semantic similarity of similar sets or words when the symbols are different.

Approach using embeddings without training on word sets Word vectors allow for soft computation of semantic similarity in word sets. In this work, we provide a formulation that allows the computation of these symbolic sets in a pre-trained word embedding space. Many previous studies attempted to represent sets and set operations in a pre-trained embedding space. BERTScore [112] approximatively models the similarity between sentences based on the cosine similarity between their tokens by regarding a sentence as a set of context-dependent token vectors. Another approach uses optimal transport to calculate the set similarity along with the alignment between set elements, such as Word Mover's Distance [60] and Word Rotator's Distance [110]. They calculate the set similarity without formulating set operations.

Recently, another work proposed [115] the formulation of set operations in pre-trained word embedding space based on fuzzy sets. It proposed Fuzzy Bag of Words, which represents a set of words as a fuzzy set using max-pooled word vectors. In this work, we represent a word set as a subspace and propose set operations based on quantum logic by computing the degree of membership. Our method is superior to fuzzy set-based methods in our experiments.

Approach using embeddings trained on word sets Many methods for learning the representation of sets have been proposed because of the wide range

of possible applications. DeepSets [111] is a typical model that has been extended by an aggregation function [83] and an attention mechanism [62]. As discussed in Section 4.4.2, our method outperformed DeepSets, which is a strong baseline. Other methods for representing sets of words include representing words as Gaussian distributions [9,101] and using boxes in the embedding space [29,100]. These previous research studies have focused on learning methods for representing sets of words, but we provide a formalization of set representation that functions as a set of words on a pre-trained embedding space without requiring such learning. We can use our formalization to compute set representation using popular general-purpose language models like BERT, which are trained on the general domain.

4.5.2 Representation Learning of Concepts

Order embedding [99] and Box embedding [29, 66, 100] are methods that embed hypernymy and hyponymy for concept representation in Euclidean space. Embedding on hyperbolic space has also been proposed as a method for embedding such relations by making good use of the geometric properties of vector space [34, 80, 95]. In contrast to these concept representation learning methods, our aim is not to learn concept representations. Rather our method provides set operations and the representation of concept sets in a pre-trained embedding space that is not specifically trained for concept representation.

4.5.3 Sentence Embeddings and Similarity

Many methods have been proposed for embedding sentences. In recent years, methods have been proposed for acquiring superior sentence representations by fine-tuning language models [19, 35, 50, 90, 113]. Although these methods are for learning sentence vectors, we proposed a novel similarity between two embedding sets. When computing the similarity of pre-trained embeddings, while cosine similarity is still used, we compute the similarity by representing a sentence as a subspace. As shown in the experiments, our proposed method outperforms cosine similarity. Although the method for representing a sentence by a subspace was proposed previously [78], our approach differs in that we calculate sentence

similarity based on set similarity using quantum logic.

4.5.4 Quantum Logic

Using quantum logic was proposed to disambiguate word meanings in search engines [107]. Other works [36, 93] used quantum logic as a constraint to learn the logical structures of a knowledge graph. We formulated the representation and operations of sets in a pre-trained embedding space based on quantum logic with an extension on set membership and set similarity.

4.6 Discussion

What are the conditions for a word set to be represented as a subspace? We believe that it is important for the embedding space to have linguistic linearity. It is known from previous research that the linguistic linearity of the embedding space is strongly related to PMI in the training corpus of the word embedding [64]. Both word2vec and GloVe are based on PMI to model the co-occurrence of words, therefore, these embedding spaces have similar linguistic properties. In fact, both of them are known to be able to calculate word analogies with word vectors. The main difference is that word2vec models co-occurrences of words within sentences, whereas GloVe also takes into account co-occurrences across the entire corpus. For the subspace to be a linguistic subspace and not just a simple subspace, it should have linguistic linearity, such as $\boldsymbol{v}_{king} + \boldsymbol{v}_{queen} \sim \boldsymbol{v}_{royal}$. If the embedding space has such linguistic linearity, $\boldsymbol{v}_{royal}$ would be contained in the subspace $\mathbb{S}_{\text{Royal}}$ spanned by $\boldsymbol{v}_{king}$ and $\boldsymbol{v}_{queen}$. The conditions for the embedding space to acquire linguistic linearity are also understood [6,7,31,38]. For example, [31] proved that the embedding space (SGNS and GloVe without reconstruction error) acquires linearity, based on the statistical properties of the training corpus characterized by co-occurrence-shifted pointwise mutual information. The above discussion is about the conditions that must be met for the subspace to represent the word set. This condition suggests the possibility of fine-tuning the subspace where the linearity of the meaning of the language holds. In other words, by learning the embedding space so that the word set is embedded linearly in the subspace, the subspace can be optimized. We mention ideas about this in Chapter 5. We

have experimentally demonstrated this, and have shown through experiments that subspaces in embedding spaces like word2vec do possess such linearity (e.g. $\boldsymbol{v}_{boy} \in \mathbb{S}_{\text{Male}}$).

4.7 Conclusion

We formulated a set representation and set operations in a pre-trained word embedding space using linear subspaces based on quantum logic to extend conventional symbolic operations with vector-based representation. Our experiments demonstrated that our proposed method represented a word set, the relation between elements and sets, and the relations between two sets.

Our proposed set operations are very versatile in embedding space and vector set similarity using set operations. Our proposal can be used as a better similarity to cosine similarity between averaged vectors, which has been widely and frequently used, and may yield better results by directly optimizing a subspace instead of an averaged vector when learning vector sets.

5 Conclusion and Future Directions

This thesis work addressed the formulation of semantic representation and its operations in an embedding space, which is important as the foundation of recent NLP. Previous studies have revealed that semantic operations such as $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} = \overrightarrow{queen}$ can be realized by adding and subtracting word vectors in the embedding space. Based on these properties of the pre-trained embedding space, this thesis work attempts to formulate a new semantic operation. Such semantic operations have the advantage of generality and are important for understanding embedding spaces. In this chapter, we conclude the thesis work and discuss future directions.

5.1 Summary

In Chapter 1, we explained the importance of the study of semantic operations. First, we reviewed the historical transition from symbolic to vector representation in the approach to semantic representation. Next, we described the approach to learning vectors of words and tokens from large data sets, which is a fundamental technology in current NLP. We then explained that semantic operations can be performed with pre-trained word vectors and that such semantic operations have two advantages.

In Chapter 2, we gave a literature review on word embedding. First of all, we introduced an approach to the vector representation of language. Vector representation can capture similarities in meaning and are more memory efficient than one-hot representation, which is a local representation of meaning. Next, we discussed the semantic computation of word vectors. We showed how the

computation of analogy and semantic composition can be achieved by adding, subtracting, and averaging word vectors. Finally, we describe a method for embedding such vectors in a language. We divided them into two categories (static and dynamic embedding). Static embedding, such as word2vec, can represent a word as a vector of a fixed dimension. Dynamic embedding, such as BERT, has succeeded in representing the polysemy of a word.

In Chapter 3, we tackled formulating a new semantic operation of inverting binary attributes. An analogy-based transfer adds or subtracts a difference vector from an input word vector to transfer it to a target word vector. Since this operation depends on the attribute value of the input word, an analogy-based transfer needs explicit knowledge. However, this knowledge cannot be developed for various words and attributes in practice. We first defined an ideal transfer function that does not use the knowledge and proposed a method that can perform a transfer without relying on the knowledge by introducing the ideal mapping called a *reflection*. A reflection is a mapping that inverts two vectors with a hyperplane called *mirror* in a Euclidean space. Thus, a reflection can transfer both the input word vector and the target word vector with the same mapping. We proposed parameterized mirror method, which dynamically predicts mirrors according to attributes and input words. In this chapter, we show that the proposed method outperforms other methods such as analogy in accuracy of word attribute transfer. Interestingly, we found that reflection-based methods are very stable when inputting words that do not have the specified attribute values.

In Chapter 4, we formulated representations and operations of word sets in the embedding space. Sets of vectors are frequently used in NLP. For example, it is often done to consider a sentence as a set of words and compute the sentence vector by averaging the word vectors. Although it is very simple and effective, it is unclear how additive composition holds properties as a set of words. However, set operations are important because they can be general-purpose tools in NLP. For example, a sentence similarity task requires the calculation of set similarity. In this chapter, we formulate a set representation and five set operations in the pre-trained word embedding space. Inspired by quantum logic, one of the theories of quantum mechanics, we give a formulation of set representation and set operation based on linear subspaces. Our experiments demonstrated that our proposed

method represented a word set, the relation between elements and sets, and the relations between two sets. We also applied the proposed method to two tasks, Text Concept Set Retrieval and Semantic Textual Similarity, and demonstrated that our proposed method has a high performance.

5.2 Contributions

In this section, we review the contributions along with the major findings of this thesis work. The main research questions and contributions are as follows.

- ***What is the significance of semantic operations in an embedding space?***
  This is our main research question in this thesis. We answered this research question through this thesis. There are two significant points in semantic operations. The first significance is that semantic operations have a wide range of applications, i.e., they are highly versatile. Semantic operations can be used with a given pre-trained embedding without specific training. For example, semantic operations can be used to calculate the similarity of sentences. Conversely, semantic operations can be used to learn the embedding space. For example, in previous research [16], an analogy was introduced to the loss function to learn the embedding space. The second significance is that it helps us understand the embedding space. The discovery of a new semantic operation will reveal one of the fundamental properties of embedding spaces. We formulated two novel semantic operations in this thesis.

- ***What is the condition for mapping to invert binary attributes between words in a pre-trained word embedding space? Which kind of mapping should be employed to satisfy such conditions?***
  We answered these research questions in Chapter 3. Considering the gender attribute transfer as an example, the ideal binary word attribute transfer f_{gender} transforms $\overrightarrow{king}$ into $\overrightarrow{queen}$ and $\overrightarrow{queen}$ into $\overrightarrow{king}$. However, $\overrightarrow{apple}$ should not be transferred by f_{gender}. In other words, the condition for f is a mapping such that $\overrightarrow{word} = f_{gender}(f_{gender}(\overrightarrow{word}))$, which is an identity

map when applied twice (subsection 3.4.1). We employed reflection, a kind of such mapping, for binary word attribute transfer (subsection 3.4.2).

- ***What are the advantages of reflection-based word attribute transfer? How has it been demonstrated?***
  We answered these research questions in Chapter 3. Reflection-based word attribute transfer has two advantages. The first advantage is that reflection does not require knowledge of the attributes of the input word. For example, in gender transformation, reflection can invert the gender of an input word regardless of whether it is male or female. Such a transformation is not possible with the analogy. The second advantage is very high stability. For example, when *apple* was input to reflection in the gender transformation, almost 100% of the words were not transformed. This is in contrast to other methods (section 3.5). These advantages allow for automatic and stable sentence creation without the use of human knowledge in sentence data augmentation.

- ***How do we represent sets and formulate set operations in prelearned embedding spaces?***
  We answered this research question in Chapter 4. Inspired by quantum logic, we proposed a formulation of sets and set operations based on linear subspaces (section 4.3). Our experimental results show that subspaces can represent linguistic set operations (e.g. *orange* $\in$ Color $\cap$ Fruit) (subsection 4.4.1).

- ***What are the advantages of subspace-based set operations? and how have they been demonstrated?***
  We answered this research question in Chapter 4. There are two advantages of set operations. The first advantage is that a combination of set operations can be used to define various computations on sets. For example, we defined the computation of sentence similarity as the similarity between word sets. The second advantage is high performance in downstream tasks (subsection 4.4.3). For example, our subspace-based sentence similarity performs better than the cosine similarity between averaged vectors. This is a highly influential result. Cosine similarity between averaged vectors

has been widely employed, for example, in the evaluation and learning of embedding spaces. This suggests that the proposed method may improve the performance of various tasks.

5.3 Future Directions

In recent years, the use of pre-trained neural networks with fine-tuning [30] and prompting [18] has become the dominant approach in the field of AI. It has proven to be effective in many tasks, such as natural language processing, computer vision, and speech recognition. The field of AI is evolving, and new methods may emerge, but the vector-based design of deep learning is likely to remain unchanged. Vector computations play a fundamental role in representing, handling, and understanding complex information of natural language, images, and speech. The idea of embedding — representing complex data in a lower-dimensional space — is a key concept in deep learning and will remain very important in the future of AI technologies.

This thesis work focused on the relationship between properties of embedding spaces and the meaning in natural language and investigated how linearity, geometry, and other properties of embedding spaces can be used to represent and compute the natural language meaning. It opens up new directions for future research in the field of NLP. Such vector operations form the foundation of deep learning and can be applied not only in NLP but in any field that uses deep learning. The semantic operations we have developed can contribute to the field of AI as a new technique and technology that can be used to improve the performance of deep learning models. They will be extended to improve the performance of various NLP tasks and open new possibilities for understanding and manipulating meaning in natural language. The semantic operation will contribute to the field of AI by providing new methods and techniques that can be used to improve the performance of deep learning models. In the remainder of this section, we explore future applications of this thesis work with some concrete ideas.

Binary Attribute Transfer Future work includes applications to other transfer tasks: sentence by sentence transfer [49, 82, 86], and entity prediction on

an analogic graph embedding space [69], in the field of computer vision, visual analogy [89], or style transfer [67, 116] with Generative Adversarial Networks (GANs) [41, 87] and Variational Auto-Encoder (VAE) [58] because their latent space holds analogic relations [87]. Since the latent space of GAN, VAE, and TransE is the space in which additive composition is valid, the proposed method can be applied.

Subspace-based Set Operations Subspaces have been applied to various technologies. For example, in the field of meta-learning, Jiang et al. [51] represented task model parameters in subspaces to efficiently represent meta-knowledge. Li et al. [65] focused on the problem of huge parameter space of neural nets and represented network parameters in low-dimensional subspaces. Subspaces are also used in learning structured data such as knowledge graphs [36, 93].

Future work includes applications to a variety of tasks. In particular, our subspace-based set similarity is widely applicable. We expect that replacing the cosine similarity between averaged vectors with our proposed set similarity will improve performance. This is because our set similarity achieved better performance than the cosine similarity between averaged vectors in our experiments. A promising application would be contrastive learning [24,35]. Contrastive learning is a method to compensate for embedding space. In contrastive learning, embeddings in pre-trained language models are fine-tuned to increase the similarity between input sentences and positive examples and to decrease the similarity between input sentences and negative examples that are not similar. In our experiments, we used a language model fine-tuned by contrastive learning (SimCSE [35]) to compute sentence similarity. Conversely, our set similarity can be used for SimCSE. SimCSE uses $[CLS]$ token as the sentence representation and the cosine similarity between $[CLS]$ tokens as the sentence similarity. A subspace can be used for a sentence representation and our subspace-based set similarity can be used for sentence similarity.

Acknowledgements

As a Ph.D. student at the Augmented Human Communication (AHC) Lab., it has been an amazing experience for me over the past few years, full of learning, growing, and developing.

First, I would like to thank Professor Satoshi Nakamura for welcoming me to AHC Lab. and supervising me for these five years. He provides a lot of insightful advice for each topic I have worked on and gave me a lot of freedom in choosing research topics, which helps me to become a well-rounded young researcher.

I would like to thank Associate Professor Katsuhito Sudoh. He has been my closest and strongest support for the past five years. There are many things I could not have done without him. He has greatly influenced me as a researcher and as a person.

I would like to thank the other members of the thesis committee, Professor Taro Watanabe and Professor Danushka Bollegala. Professor Watanabe gave me valuable comments and suggestions that greatly improved my thesis. Professor Bollegala accepted me for an internship at the University of Liverpool. Unfortunately, I was not able to study there in the UK, but he gave me insightful ideas and new knowledge during our online meetings.

I would like to thank Assistant Professor Sho Yokoi. I am grateful for the inspiring discussions and the exchange of ideas during our collaboration with him. His proactive advice greatly improved my skills as a researcher.

I would like to express my gratitude to all the faculty I have worked with in the AHC Lab., for their support in every aspect of my research life.

Finally, I would like to thank my parents, brother, and friends for supporting my Ph.D. and my life.

Bibliography

[1] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In Daniel M. Cer, David Jurgens, Preslav Nakov, and Torsten Zesch, editors, *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 252–263. The Association for Computer Linguistics, 2015.

[2] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In Preslav Nakov and Torsten Zesch, editors, *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*, pages 81–91. The Association for Computer Linguistics, 2014.

[3] Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In Steven Bethard, Daniel M. Cer, Marine Carpuat, David Jurgens, Preslav Nakov, and Torsten Zesch, editors, *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 497–511. The Association for Computer Linguistics, 2016.

[4] Eneko Agirre, Daniel M. Cer, Mona T. Diab, and Aitor Gonzalez-Agirre. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In Eneko Agirre, Johan Bos, and Mona T. Diab, editors, *Proceedings of the 6th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2012, Montréal, Canada, June 7-8, 2012*, pages 385–393. The Association for Computer Linguistics, 2012.

[5] Eneko Agirre, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *SEM 2013 shared task: Semantic Textual Similarity. In Mona T. Diab, Timothy Baldwin, and Marco Baroni, editors, *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA*, pages 32–43. Association for Computational Linguistics, 2013.

[6] Carl Allen and Timothy M. Hospedales. Analogies Explained: Towards Understanding Word Embeddings. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 223–231, 2019.

[7] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A Latent Variable Model Approach to PMI-based Word Embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

[8] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[9] Ben Athiwaratkun and Andrew Gordon Wilson. Multimodal Word Distributions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1645–1656, 2017.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio

and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[11] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 932–938. MIT Press, 2000.

[12] Garrett Birkhoff and John Von Neumann. The logic of quantum mechanics. *Annals of mathematics*, pages 823–843, 1936.

[13] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[14] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[15] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

[16] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-

relational Data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795, 2013.

[17] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642. The Association for Computational Linguistics, 2015.

[18] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[19] Fredrik Carlsson, Amaru Cuba Gyllensten, Evangelia Gogoulou, Erik Ylipää Hellqvist, and Magnus Sahlgren. Semantic Re-tuning with Contrastive Tension. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[20] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan,

Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal Sentence Encoder. *CoRR*, abs/1803.11175, 2018.

[21] Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In Steven Bethard, Marine Carpuat, Marianna Apidianaki, Saif M. Mohammad, Daniel M. Cer, and David Jurgens, editors, *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 1–14. Association for Computational Linguistics, 2017.

[22] Noam Chomsky. *Syntactic Structures*. Mouton and Co., The Hague, 1957.

[23] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311, 2022.

[24] Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Solja$\boldsymbol{v}_c$ić, Shang-Wen Li, Wen-tau Yih, Yoon Kim, and James Glass. Diffcse: Difference-based contrastive learning for sentence embeddings. *arXiv preprint arXiv:2204.10298*, 2022.

[25] Kenneth Ward Church and Patrick Hanks. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[26] Allan M Collins and M Ross Quillian. Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior*, 8(2):240–247, 1969.

[27] Alexis Conneau and Douwe Kiela. SentEval: An Evaluation Toolkit for Universal Sentence Representations. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Kôiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA), 2018.

[28] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680. Association for Computational Linguistics, 2017.

[29] Shib Sankar Dasgupta, Michael Boratko, Siddhartha Mishra, Shriya Atmakuri, Dhruvesh Patel, Xiang Li, and Andrew McCallum. Word2Box: Capturing Set-Theoretic Semantics of Words using Box Embeddings. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2263–2276. Association for Computational Linguistics, 2022.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors,

Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics, 2019.

[31] Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. Towards Understanding Linear Word Analogies. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3253–3262, 2019.

[32] Charles J Fillmore. The case for case. 1967.

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

[34] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic Entailment Cones for Learning Hierarchical Embeddings. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1632–1641. PMLR, 2018.

[35] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics, 2021.

[36] Dinesh Garg, Shajith Ikbal, Santosh K. Srivastava, Harit Vishwakarma, Hima P. Karanam, and L. Venkata Subramaniam. Quantum Embedding of Knowledge for Reasoning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual*

Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 5595–5605, 2019.

[37] Zoubin Ghahramani and Katherine A. Heller. Bayesian Sets. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 435–442, 2005.

[38] Alex Gittens, Dimitris Achlioptas, and Michael W. Mahoney. Skip-Gram - Zipf + Uniform = Vector Additivity. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 69–76, 2017.

[39] Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the Student Research Workshop, SRW@HLT-NAACL 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 8–15, 2016.

[40] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 315–323. JMLR.org, 2011.

[41] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[42] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. OpenKE: An Open Toolkit for Knowledge Embedding. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 139–144. Association for Computational Linguistics, 2018.

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

[44] Geoffrey E Hinton, James L McClelland, David E Rumelhart, et al. *Distributed representations*. Carnegie-Mellon University Pittsburgh, PA, 1984.

[45] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[46] W John Hutchins. The georgetown-ibm experiment demonstrated in january 1954. In *Conference of the Association for Machine Translation in the Americas*, pages 102–114. Springer, 2004.

[47] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015.

[48] Paul Jaccard. Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:241–72, 01 1901.

[49] Parag Jain, Abhijit Mishra, Amar Prakash Azad, and Karthik Sankaranarayanan. Unsupervised Controllable Text Formalization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, pages 6554–6561, 2019.

[50] Ting Jiang, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen Huang, Liangjie Zhang, and Qi Zhang. Prompt-BERT: Improving BERT Sentence Embeddings with Prompts. *CoRR*, abs/2201.04337, 2022.

[51] Weisen Jiang, James Kwok, and Yu Zhang. Subspace learning for effective meta-learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 10177–10194. PMLR, 2022.

[52] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[53] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[54] Masahiro Kaneko and Danushka Bollegala. Gender-preserving Debiasing for Pre-trained Word Embeddings. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1641–1650, 2019.

[55] Dongyeop Kang, Tushar Khot, Ashish Sabharwal, and Eduard H. Hovy. AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2418–2428. Association for Computational Linguistics, 2018.

[56] Joo-Kyung Kim and Marie-Catherine de Marneffe. Deriving Adjectival Scales from Continuous Space Word Representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1625–1630. ACL, 2013.

[57] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[58] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[59] Andrew V. Knyazev and Merico E. Argentati. Principal angles between subspaces in an A-based scalar product: Algorithms and perturbation estimates. *SIAM J. Sci. Comput.*, 23(6):2008–2040, 2002.

[60] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. From Word Embeddings To Document Distances. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 957–966. JMLR.org, 2015.

[61] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-Based & Neural Unsupervised Machine Translation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 5039–5049. Association for Computational Linguistics, 2018.

[62] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR, 2019.

[63] Omer Levy and Yoav Goldberg. Linguistic Regularities in Sparse and Explicit Word Representations. In *Proceedings of the Eighteenth Conference*

on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014, pages 171–180, 2014.

[64] Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185, 2014.

[65] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[66] Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. Smoothing the Geometry of Probabilistic Box Embeddings. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[67] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Trans. Graph.*, 36(4):120:1–120:15, 2017.

[68] Tal Linzen. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, RepEval@ACL 2016, Berlin, Germany, August 2016*, pages 13–18, 2016.

[69] Hanxiao Liu, Yuexin Wu, and Yiming Yang. Analogical Inference for Multi-relational Embeddings. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2168–2178, 2017.

[70] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692, 2019.

[71] Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 5108–5118, 2018.

[72] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 2001.

[73] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, pages 216–223. European Language Resources Association (ELRA), 2014.

[74] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[75] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

[76] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 746–751, 2013.

[77] George A. Miller. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41, 1995.

[78] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. Representing sentences as low-rank subspaces. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 629–634. Association for Computational Linguistics, 2017.

[79] Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 76–85, 2017.

[80] Maximilian Nickel and Douwe Kiela. Poincaré Embeddings for Learning Hierarchical Representations. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6338–6347, 2017.

[81] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A Three-Way Model for Collective Learning on Multi-Relational Data. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 809–816. Omnipress, 2011.

[82] Xing Niu, Sudha Rao, and Marine Carpuat. Multi-Task Neural Models for Translating Between Styles Within and Across Languages. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1008–1021, 2018.

[83] Giovanni Pellegrini, Alessandro Tibo, Paolo Frasconi, Andrea Passerini, and Manfred Jaeger. Learning Aggregation Functions. In Zhi-Hua Zhou,

editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 2892–2898. ijcai.org, 2021.

[84] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, 2014.

[85] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

[86] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W. Black. Style Transfer Through Back-Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 866–876, 2018.

[87] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[88] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[89] Scott E. Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep Visual Analogy-Making. In *Advances in Neural Information Processing Systems*

28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pages 1252–1260, 2015.

[90] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics, 2019.

[91] Radu Soricut and Franz Josef Och. Unsupervised Morphology Induction Using Word Embeddings. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1627–1637, 2015.

[92] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[93] Santosh K. Srivastava, Dinesh Khandelwal, Dhiraj Madan, Dinesh Garg, Hima Karanam, and L. Venkata Subramaniam. Inductive quantum embedding. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[94] Md. Arafat Sultan, Steven Bethard, and Tamara Sumner. DLS$@$CU: Sentence Similarity from Word Alignment and Semantic Vector Composition. In Daniel M. Cer, David Jurgens, Preslav Nakov, and Torsten Zesch, editors, *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 148–153. The Association for Computer Linguistics, 2015.

[95] Ryota Suzuki, Ryusuke Takahama, and Shun Onoda. Hyperbolic Disk Embeddings for Directed Acyclic Graphs. In Kamalika Chaudhuri and Ruslan

Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6066–6075. PMLR, 2019.

[96] Vikas Thada and Vivek Jaglan. Comparison of jaccard, dice, cosine similarity coefficient to find best fitness value for web retrieved documents using genetic algorithm. *International Journal of Innovations in Engineering and Technology*, 2(4):202–205, 2013.

[97] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.

[98] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[99] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-Embeddings of Images and Language. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[100] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018,*

Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, pages 263–272. Association for Computational Linguistics, 2018.

[101] Luke Vilnis and Andrew McCallum. Word Representations via Gaussian Embedding. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[102] Oriol Vinyals and Quoc V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015.

[103] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275, 2019.

[104] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[105] Donald A Waterman. *A guide to expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1985.

[106] Warren Weaver. Translation. In *Proceedings of the Conference on Mechanical Translation*, 1952.

[107] Dominic Widdows. Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In Erhard W. Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan*, pages 136–143. ACL, 2003.

[108] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[109] Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Unsupervised Neural Machine Translation with Weight Sharing. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 46–55. Association for Computational Linguistics, 2018.

[110] Sho Yokoi, Ryo Takahashi, Reina Akama, Jun Suzuki, and Kentaro Inui. Word Rotator's Distance. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2944–2960. Association for Computational Linguistics, 2020.

[111] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep Sets. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3391–3401, 2017.

[112] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[113] Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. An Unsupervised Sentence Embedding Method by Mutual Information Maxi-

mization. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1601–1610. Association for Computational Linguistics, 2020.

[114] Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. Learning Gender-Neutral Word Embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4847–4853, 2018.

[115] Vitalii Zhelezniak, Aleksandar Savkov, April Shen, Francesco Moramarco, Jack Flann, and Nils Y. Hammerla. Don't Settle for Average, Go for the Max: Fuzzy Sets and Max-Pooled Word Vectors. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[116] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251, 2017.

Publication List

Journal Paper (peer-reviewed)

1. **Yoichi Ishibashi**, Katsuhito Sudoh, Koichiro Yoshino, Satoshi Nakamura. Reflection-based Word Attribute Transfer. In *Journal of Natural Language Processing, vol.28, no.1, Mar. 2021.*
   (Related to Chapter 3)

International Conference Paper (peer-reviewed)

1. **Yoichi Ishibashi**, Danushka Bollegala, Katsuhito Sudoh, Satoshi Nakamura. Evaluating the Robustness of Discrete Prompts. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023), May 2023.*

2. **Yoichi Ishibashi**, Katsuhito Sudoh, Koichiro Yoshino, Satoshi Nakamura. Reflection-based Word Attribute Transfer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020): Student Research Workshop, 51 - 58, July. 2020.*
   (Related to Chapter 3)

Preprint

1. **Yoichi Ishibashi**, Sho Yokoi, Katsuhito Sudoh, Satoshi Nakamura. Subspace-based Set Operations on a Pre-trained Word Embedding Space. *Preprint (Submitted to IEEE Access), Oct. 2022.*
   (Related to Chapter 4)

Domestic Conferences

1. 石橋陽一, 横井祥, 須藤克仁, 中村哲. 正準角および部分空間に基づく BERTScore の拡張, 言語処理学会第 29 回年次大会 (NLP 2023), Mar. 2023.

2. 石橋陽一, 横井祥, 須藤克仁, 中村哲. 学習済み埋め込み空間の線型部分空間を用いた集合演算, 第 36 回人工知能学会全国大会 (JSAI 2022), Jun. 2022.

3. 石橋陽一, 横井祥, 須藤克仁, 中村哲. 線型部分空間に基づく学習済み単語埋込空間上の集合演算, 言語処理学会第 28 回年次大会 (NLP 2022), Mar. 2022. 若手奨励賞

4. 石橋陽一, 横井祥, 須藤克仁, 中村哲. 学習済み埋め込み空間における集合演算, 第 24 回情報論的学習理論ワークショップ (IBIS 2021), Nov. 2021.

5. 石橋陽一, 横井祥, 須藤克仁, 中村哲. 量子論理に基づく単語埋込集合間の論理演算, NLP 若手の会第 16 回シンポジウム (YANS 2021), Aug. 2021.

6. 石橋陽一, 須藤克仁, 中村哲. 単語属性変換による自然言語推論データの拡張, 言語処理学会第 27 回年次大会 (NLP 2021), Mar. 2021.

7. 石橋陽一, 須藤克仁, 吉野幸一郎, 中村哲. 鏡映変換に基づく埋め込み空間上の単語属性変換, 言語処理学会第 26 回年次大会 (NLP 2020), Mar. 2020. 優秀賞

8. 石橋陽一, 須藤克仁, 吉野幸一郎, 中村哲. 鏡映変換に基づく埋め込み空間上の単語属性変換, 情報処理学会研究報告, Vol. 2019-NL-241, No. 9, pp. 1-7, 2019. 優秀研究賞

9. 石橋陽一, 須藤克仁, 吉野幸一郎, 中村哲. 鏡映変換に基づく埋め込み空間上の単語属性変換, NLP 若手の会第 14 回シンポジウム (YANS 2019), Aug. 2019. 奨励賞