Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Xinyi Jiang March 25, 2019

Incremental sense weight training for contextualized word embedding interpretation

by

Xinyi Jiang

Jinho D. Choi
Adviser

Department of Computer Science

Jinho D. Choi

Adviser

Justin C. Burton

Committee Member

Davide Fossati

Committee Member

2019

Incremental sense weight training for contextualized word embedding interpretation

By

Xinyi Jiang

Jinho D. Choi

Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Computer Science

2018

Abstract

Incremental sense weight training for contextualized word embedding interpretation

By Xinyi Jiang

In this work, we propose a new training procedure for learning the importance of dimensions of word embeddings in representing word meanings. Our algorithm advanced in the interpretation filed of word embeddings, which are extremely critical in the NLP filed due to the lack of understanding of word embeddings despite their superior ability in progressing NLP tasks. Although previous work has investigated in the interpretability of word embeddings through imparting interpretability to the embedding training models or through post-processing procedures of pre-trained embeddings, our algorithm proposes a new perspective to word embedding dimension interpretation where each dimension gets evaluated and can be visualized. Also, our algorithm adheres to a novel assumption that not all dimensions are necessary for representing a word sense (word meaning) and dimensions that are negligible get discarded, which have not been attempted in previous studies.

Incremental sense weight training for contextualized word embedding interpretation

By

Xinyi Jiang

Jinho D. Choi

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Computer Science

2018

Acknowledgements

I would first like to thank my parents for all the help and sponsor that I have received from them. It is with their help that I have been able to study and mature in Emory University.

I would like to thank all of my committee members for having supported, guided and mentored me throughout my career at Emory. I am grateful for all the help that I am so lucky to have received from them. Dr. Fossati is the first to introduce me into the field of Computer Science and Dr.Burton is the one who guided me into the journey of research. I would also like to give special thanks to my advisor, Jinho D. Choi, as his guidance helped me through some of my most difficult moments in Emory and this work would never be possible without his help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Word embeddings, representations of words produced based on their distributional semantics and context, has been an valuable tool utilized in various NLP tasks. With the training of relevant models on a large-size corpus, word embeddings capture characteristics of words in dense vectors with rich semantic and syntactic information. Such representations will later be used in the downstream Natural Language Processing (NLP) tasks, which advance the progress of other NLP tasks compared to sparse vector of Bag-Of-Word methods. However, the dense quality of embeddings has also made it hard to interpret meanings associated with each dimension, making word embeddings black-box-like. Also, with the development of contextual word embedding models, the word embedding dimension size has increased from around 300 to 2000, due to the performance gain caused by the increase of dimensions size. As a result, the work of interpreting word embeddings has become more critical

as more factors are involved. In this work, we propose a novel algorithm for learning and understanding the value of word embedding dimensions. The following sections give an overview of the problem description, motivation, objectives and related works relating to this thesis.

1.1 Problem Description

There has not yet been a consensus on the evaluation and interpretation of word embeddings. There are mainly two types of assessments: intrinsic evaluations and extrinsic evaluations. Intrinsic evaluations often investigate the nature of semantics by comparing embedding relations and human cognitive sciences, which include both conscious and subconscious sciences [3, 26, 7]. Meanwhile, extrinsic evaluations focus on the engineering perspective where word embeddings are used as input to downstream tasks and evaluate the performance of embeddings on specific tasks [3]. On the other hand, the task of imparting interpretability has been tackled in various ways in the literature. Some researchers modified the training processes of embeddings to encourage interpretable dimension embedding learning [24, 20], while others suggested mapping from uninterpretable embedding to embeddings with more interpretable dimensions [2, 10].

Although the approaches mentioned above provide better understandings and interpretations to the word embeddings, many intrinsic evaluations have become ineffective due to the emergence of context-based word embeddings, which required contextual information as input. Moreover, we want to explore the possibilities of interpreting word embedding dimensions directly rather than trying to modify the current dimensions. This thesis aims to evaluate and understand context-based word embedding, mainly in the field of word sense disambiguation. We offer a hypothesis that some dimensions do not play a role in representing a word sense. In this work, we propose an algorithm to analyze and visualize the functionalities of specific dimensions to word meanings.

1.2 Motivation

There are three main reasons to work on the task of word embedding evaluation and interpretation. First, word embeddings are widely used in almost all NLP tasks, and as already illustrated in previous works, the quality of meaningful representations of words have a big impact on the performances of models for NLP tasks [36, 1, 6]. Considering the importance of word embeddings, it is intrinsically intriguing in investigating what information is being captured in

the embeddings.

Second, we plan to aim our project direction to word sense detailed evaluations. With emerging context-based word embeddings which take sentences as inputs, embeddings are highly contextualized. Based on the knowledge that humans understand language and distinguish senses based on the contextual information implied, we would like to investigate in how much sense contextual information is contained in the word embeddings.

Third, although embedding models usually offer options to generate embeddings with a wide range of word embedding dimension sizes, there has not been full understandings of the functionality of specific dimensions. Most studies in the word embedding interpretability field have been done using either statistical methods or modifications to the embedding training procedures to generate more interpretable word embeddings [51, 20]. However, in this work, we want to approach the problem of interpretability from a different perspective to have the machine learn to understand specific dimensions in word embeddings.

1.3 Objectives

The primary goal of this thesis is to advance the research in the word embedding evaluation and interpretation field. The concrete objectives are listed below:

- evaluate the ability to learn sense information by analyzing sense groups and similarity tests of context-based word embedding models.
- impart interpretability to embedding dimensions by proposing a new algorithm for determining the importance of word embedding dimensions
- evaluate the algorithm by masking out unimportant dimensions of word embeddings and testing the properties of masked word embeddings

1.4 Related Works

In this section, we will introduce works that have been done to impart interpretability to word embeddings. In the early works, Murphy et al.(2012) suggested a variant of sparse matrix factorization, called Non-Negative Sparse Embedding (NNSE), which can generate highly interpretable word representations [29]. Based on NNSE, Jang and Myaeng introduced a method analyzing dimensions characterizing categories by linking concepts with types

using HyperLex datasets [48] and comparing dimension values within concept groups with the average of dimension values within category groups [17]. Matrix factorization techniques were also used to extend Skip-Gram models [26] by applying a projected gradient from non-negative matrix factorization (NMF) [22] to modify the learning process of Skip-Gram [24]. Thus the modified model encourages learning of interpretable word embeddings since NMF only forbid combinations of addition operations over subtraction operations, which result in an embedding that is part-based [24]. Other work [20] also invested in the interpretability topic by extending current embedding models of Glove [34]. Via mapping group label information obtained from Rogets Thesaurus, the words belonging to certain groups, are encouraged to learn an elevated amount in the dimensions that correspond to the group [20]. Other works make use of pre-trained embeddings and apply post-processing techniques to acquire embeddings with more interpretability. Researches [53, 33] used matrix transformation methods on pre-trained embeddings. The approach in [53] utilized canonical orthogonal transformations to map current embeddings to a new vector space where the meanings of components are more interpretable. Similar to the work of [53], Park, Bak and Oh(2017) proposed an approach that rotates pre-trained embedding by minimizing

the complexity function so that the dimensions after rotation become more interpretable [33]. Another type of methods apply sparse encoding techniques on word embeddings and map them to sparse vectors [45, 2]. [43] directly analyzes dimensions of embeddings using eigenvector analysis.

Chapter 2

Background

2.1 Word embedding extraction

Word embedding extraction has been one of the leading research topics in natural language processing. By using language modeling and feature learning techniques, including neural networks, words get mapped to vectors, which are called word embeddings. Based on the idea that "a word is characterized by the company it keeps" [13], word embeddings aim to incorporate more syntactic and semantic information by training on large natural language corpora. Different from the Vector Space Model [21] before, word embeddings are dense vectors with a much smaller dimension size. Popular word embedding models include Word2Vec [26], FastText [18], GloVe [34]. All Word2Vec, FastText and GloVe take words as inputs and perform mapping in a dictionary way where one word has a definite embedding stored and programs access

embeddings by looking up the word in the dictionary. These models prose a problem where each word only has one embedding while polysemy is common, where a word has multiple meanings. Context-based models: ELMo [36], Flair [1] and BERT [6] models, on the contrary, produce an embedding for each occurrence of a word. Sentences are taken as inputs in this case, and the output word embedding is highly contextualized.

2.2 Embedding evaluations

Embedding evaluations can be divided into two main categories: intrinsic evaluations and extrinsic evaluations. Intrinsic evaluations investigate in the nature of semantics, usually through comparison with embedding relationships and human cognitive sciences while extrinsic evaluations tend to apply embeddings to downstream tasks and evaluate the performance of embeddings on specific tasks. The following two sections will introduce widely-used intrinsic evaluation methods and extrinsic evaluation methods separately.

2.2.1 Intrinsic evaluations

Intrinsic evaluations focus mainly on word embedding intrinsic qualities as well as how well embedding relations relate to human cognitions.

Word semantic similarity test is one of the most widely used intrinsic evaluations. Datasets were collected from human evaluators, who, given pairs of words, were asked to access the similarity of two words within a pair. Similarities calculated by distance within word embeddings of word pairs from the same datasets were compared with the human judgement results. Many datasets have been developed for similarity tests, including WordSim-353 [12] and SimVerb-3500 [15].

Another popular intrinsic evaluation is word analogy test, which was popularized by the word2vec word embedding model [26]. The main assumption is that arithmetic operations on word embeddings can solve analogy problems. Meanly, given four words x,x', y and y', if the relationship between x and x' is the same as the relationship between y and y', it is believed that y' can be predicted by $x + x' - y$. One popular example would be given the embeddings of King, Man, Queen, then Women embedding can be predicted by king + man - queen. Popular datasets for analogy tests contain Google Analogy [12] and SemEval-2012 [19].

Other intrinsic evaluations include subconscious intrinsic evaluations. Subconscious intrinsic evaluations are evaluations where human cognitive assessments are caught in a subconscious level. For instance, eye movement

data evaluations compare the features extracted from the eye movement data and word pair embeddings [44]. Subconscious evaluations also include semantic priming, which is based on the assumption that if a word is preceded by a semantically related word, then a human will tend to read the word faster. Word semantic relations are evaluated based on the time people spent reading the word, given the previous one [7].

2.2.2 Extrinsic evaluations

Extrinsic evaluations measure the capability of word embeddings to be applied as the input vector for machine learning algorithms in downstream NLP tasks. In theory, word embeddings can be used in almost any NLP tasks. For example, the General Language Understanding Evaluation (GLUE) benchmark is one of the popular framework for extrinsic evaluations, which is a collection of various natural language understanding tasks, with relevant datasets [49].

2.2.3 Intrinsic and extrinsic evaluations

Several studies have also investigated in the relationship between intrinsic evaluation and extrinsic evaluation performances of word embeddings [40, 25, 5]. Rogers et al. (2018) included more intrinsic properties of words extracted from sources of WordNet [11], Wiktionary, and BabelNet [30] in evaluation

relationship experiments, compared to the previous works.

2.3 Sense Vector extraction

A sense is one of the meanings of a word. For humans, the contexts of a word hint us which sense it is in particular sentences. Thus, word senses and concepts play an essential role in linguistic understanding. The task of sense vector extraction is to generate a meaningful representation for each sense. The early stages of works explored clustering techniques [42, 16, 39] before assigning of sense representations. Other techniques that adopt distributional models from word embeddings to sense embeddings have also been explored. Neelakantan et al. (2015) was the first to extend a word embedding model, Skip-Gram model [26] to develop a multi-sense model [31]. The model was introduced as Multiple-Sense Skip-Gram. In this method, the contexts of a word are represented as a centroid for clustering and word vector gets updated as the clustering groups get updated with the new addition of contexts[31]. Later works [23, 32] further incorporate topic information to the training of embeddings.

Another significant section of embedding extraction is based on outside sources such as WordNet [11] and Wikipedia. Models integrate additional

semantic information of senses from lexical resources during training [46, 50]. Using word sense graph WordNet provides, other models use random graph walks on WordNet synsets (units that represent senses) and compute a representation for each synset [37, 38]. Rothe and Schtze (2014) proposed a different method for generating synset embeddings in the same vector space as the word embeddings by utilizing an autoencoding framework [41]. Faruqui el at. (2015) introduced a method called retrofitting [9] where during training based on semantic graph is to encourage similarity between a words vector and vectors with senses connected in the graph and the method of retrofitting is further utilized in [8, 9].

2.4 Neural Networks

2.4.1 Convolutional Neural Network

Convolutional Neural Network (CNN) was initially used for image processing and were responsible for major breakthroughs in image classification. Through a series of convolution and pooling operations, CNN selects specific features and end up keeping the most salient ones. When CNN gets applied to NLP, convolution operations usually slide over full rows of the input matrix. By doing so, features such as n-grams information can get learned easily by CNN.

2.4.2 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) networks are a specific type of Recurrent Neural Networks (RNN), designed for long-term dependencies. RNN is an artificial neural network with nodes connected along a temporal sequence. Thus RNN nodes can retain information from previous nodes, thus capturing sequential information. LSTM networks are a specific type of RNN where cells are gated to control the flow of information. As humans read languages in a timely manner, it seems only natural to consider sequential information in NLP tasks. As a result, applications of LSTM in NLP tasks have exhibited great successes.

2.4.3 Transformer

The paper "*Attention is All You Need*" [47] first proposed the idea of the transformer. It is a neural network architecture based on an attention mechanism. The attention mechanism is used as a way for the model to focus on relevant information based on what it is currently processing.

Chapter 3

Qualitative Evaluation of Contextual Word Embeddings

During our journey in the exploration of word embeddings, we have pursued in multiple directions and in this section, approaches that have been undertook will be elaborated.

3.1 Embedding Model Structures

This work explores three popular word embedding algorithms with various dimensions: ELMo [36], Flair [1] and BERT [6]. ELMo model is learned functions of a deep word-level bidirectional LSTM language model with character level convolution networks along with a linear projection output layer [36], as displayed in the lower half of figure 3.1 before the embedding is put into the sequence tagging model. FLAIR model is character-level bidirectional LSTM language model on sequences of characters [1]. The hidden

state after the token's last character in the forward layer is concatenated with the state before the token's first character in the backward layer to generate the final output of the token's word representation [1]. Figure 3.2 shows the extraction for contextual embedding for the word Washington" in the network. BERT has a model architecture of a multi-layer bidirectional transformer encoder [6], shown in figure 3.3b. Figure 3.3a presents the structure of a transformer, which makes use of attention mechanisms and learns contextual relations between words.

3.2 Evaluations on Conversational Dataset

3.2.1 Dataset

The conversational dataset that is utilized is the Friends TV show transcript[1]. The Friends TV show transcript is a multiparty dialogue data with speaker name annotated, which contains the scripts from the ten seasons of the Friends TV show. Each season contains 24 episodes with around 6,000 utterances. The corpus, in total, includes 67,373 utterances, 126,059 sentences and 1,110,936 tokens. The training, validation and test set were split on a 19:2:3 ratio where for each season, the first 19 episodes were used as the training set, the 20 and

[1]https://github.com/emorynlp/character-mining

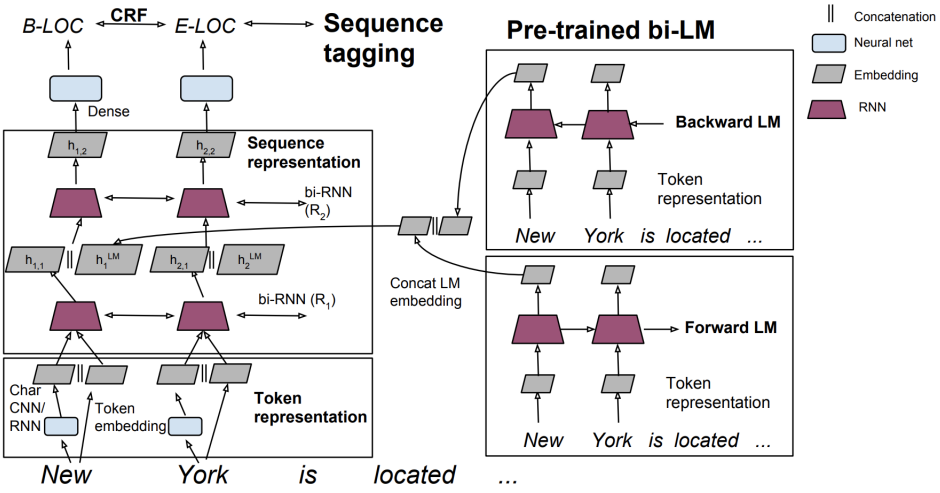


Figure 3.1: Overview of a sequence tagging model with the bottom part representing ELMo embedding structure (before the top bi-RNN networks are applied) [35]

.

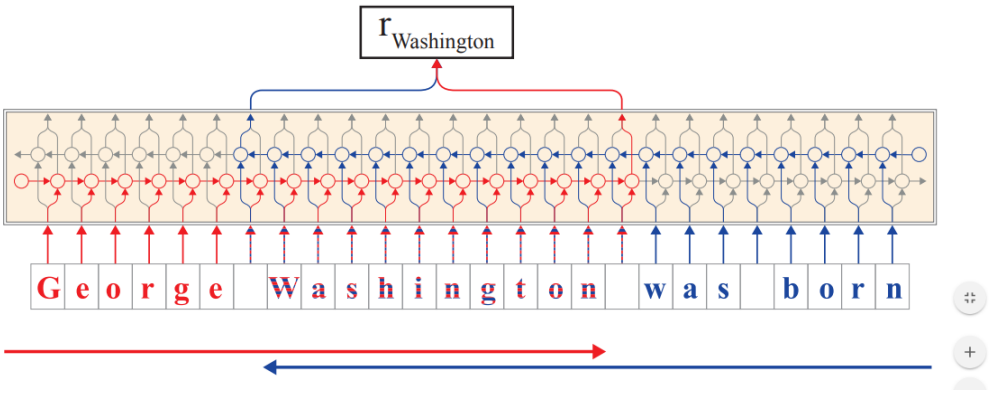
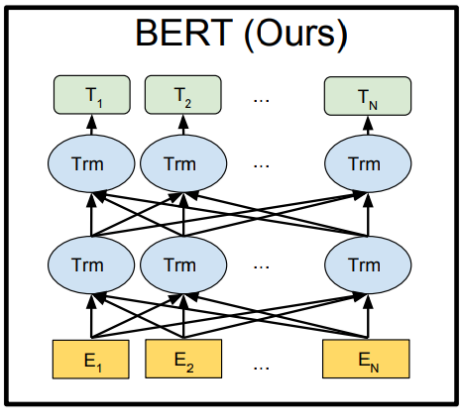
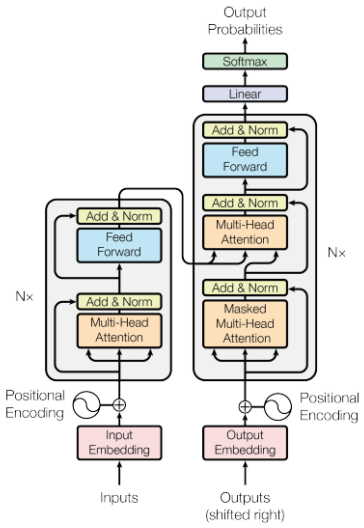


Figure 3.2: Extraction of a contextual string embedding for a word (Washington) in a sentential context in Flair model. From the forward language model (shown in red), output hidden state is extracted after the last character in the word. From the backward language model(shown in blue), the output hidden state is extracted before the first character in the word. Both output hidden states are concatenated to form the final embedding [1]

.



(a) BERT model architecture [6]



(b) The architecture of a transformer [47]

Figure 3.3: BERT model structure

21 episodes as the validation set and the last 3 episodes as the test set.

3.2.2 Embedding Models Used

The test on conversational datasets involved two embeddings: ELMo and Flair. The ELMo model has a structure of 2 layers of bidirectional LSTM with a hidden state size of 1028 and an output projection size of 128. Concatenating the output of the forward layer and backward layer results in an output embedding dimension size of 256. The average embedding of the three output layers were used. This approach also worked with the Flair model of 1 layer of bidirectional LSTM with a hidden size of 1028, producing

an output embedding dimension size of 2048. The models were prepared two ways on our dataset: trained and tuned. Trained models were directly trained on the conversational dataset, and tuned model are word embedding models originally trained on a large dataset (1-billion-word corpus) and tuned (continue training) on our conversational dataset.

3.2.3 Approach

The first evaluation method that we have tested on contextualized word embeddings were qualitative evaluations of contextualized word embeddings on conversational datasets. In our approach, we generated context-based word embeddings of Flair models and ELMo models and calculated the cosine similarity score between every pair of tokens in the corpus. Afterwards, the pair-wise cosine similarity scores of all possible pairs of words are calculated and ranked. The pairs that are at the top are later extracted. One additional test done was affinity propagation clustering on both of the flair models to see if meaningful clusters are able to be grouped.

3.2.4 Result

A part of the results for the similarity tests are shown in Table 3.1, which included the word pairs that have the highest embedding similarities and

excluded interjection word pairs. One common pattern that we noticed in the similarity rankings is that for all the models, the interjection word pairs take up a large proportion in the top rankings. Meanly, words that show emotions, such as “ahhhhh” and “ohhhhh”. The main reason that they get ranked high in the similarity test is that they have a high frequency of appearance in the conversational dataset and due to their simple morphological structure, same characters lead to more similar word embeddings. The next pattern that we noticed is that the tuned embeddings have a more stable word embedding pairs in the top, where words with similar semantic properties appear together while in the trained models, word pairs with similar morphological structures are ranked high on the list. For example, in the Flair trained model, the “preparation presentation” word pair gets ranked with highest similarity which is not that similar in word semantic meaning while “realise realize” appears in the top rankings of the tuned model, which have a closer semantic relation. In order to test if the tuned models are highly influenced by the word morphology, we applied spearman’s correlation test on the Levenshtein distance of word pairs and their cosine similarity scores. The correlation coefficient resulted in 0.000748 and 0.001497 for the Flair trained model and ELMo trained model accordingly. Thus, the conclusion can not be drawn that the word embedding

model trained on small conversational dataset is heavily influenced by the word morpogoical properties. Another pattern that we noticed is that the ELMo model tends to be able to pick out name relations as the top pairs compared to the Flair models. Table 3.2 shows some clusters in the affinity propagation cluster results on Flair models. It can be seen that both the trained word embedding model and tuned model can capture the morphology information well where words with the same suffix such as “ing” and “s” get clustered together and the tuned model have a tendency to have clusters with common topics such as ”far Over over down past around” in the table, which contain relative location information and were not necessarily alike in their forms. In general, the results of this test corresponds to what people have been believing in that when word embedding models get trained on a large corpus, the model is more generalized and robust in their applications. We can see from the experiment results that common information is being retained in the word embeddings and in the next section, we will take an approach in word embedding dimension perspective to investigate how the dimensions affect the common information in the embeddings.

	Trained	Tuned
Flair	preparation presentation 2 1 explanation explosion connection collection intimidating interesting description decision confidence coincidence	1 2 realise realize he she could can are were They You somebody someone
ELMo	Barber Farber Tribbiani Trrrribbiani Yeller Geller Lou You The She Bing Ring friend girlfriend	a an why Why Where where Of of The the What what If if

Table 3.1: top ranking results for similarity tests

	Flair
Trained	forgot forgive follow forgotten forget whoever however whenever whatever really finally talking telling thinking calling taking
Tuned	gets announces brings starts takes saves wins goes pulls choosing trying hoping deciding trying working starting far Over over down past around travel journey walk ride

Table 3.2: some clusters in the affinity propagation clustering results

Chapter 4

Representation of Sense Dimensions in Word Embeddings

4.1 Models used

In this work, we used pre-trained models provided by ELMo, Flair and BERT. We experimented on several ELMo model: model with 2 layers of bidirectional LSTM with a hidden state size of 1028 and an output projection size of 128; model with 2 layers of bidirectional LSTM with a hidden state size of 2048 and an output projection size of 256; model with 2 layers of bidirectional LSTM with a hidden state size of 4096 and an output projection size of 512 [36]. The resulting output embedding size is twice of the output size

since the output of forward-direction network and backward-direction are stacked to form the final output embedding, resulting in word embedding dimensions of 256, 512 and 1024. Since the output of ELMo model is a linear combination of three states in the neural network states and our experiments did not contain a downstream task to tune the combination, we tested the output of given by the three states separately and the average of three states output. The three states are the output of the convolutional network, the output of the first LSTM layer and the output of the second LSTM layer. All of the ELMo models were trained on 1 Billion Word Language Model Benchmark which contains approximately 800M tokens crawled from news data [36, 16].

BERT models with embedding dimension of 768 and 1024 were investigated. The first model has 12 transformer layers with a hidden dimension of 768, and attention head size of 12 and second has 24 transformer layers with a hidden size of 1024 and attention head size of 16 [1]. Both models were trained on Wikipedia (2.5B words) with BookCorpus (800M words) over a long time with 1M update steps [1, 52]. In this work, we used the output of the second to the last layer as the output embedding.

Flair models with dimension size of 2048 and 4096 were included in this

work as well. Both were trained on a 1-billion word corpus [4] with both being one layer bidirectional LSTM model.

4.2 Datasets

Datasets used in the experiments include SemCor [27] and SCWS [16]. SemCor is a semantically annotated corpus where the tokens in the corpus were manually annotated with WordNet 1.6 senses [11]. SemCor corpus consists of 352 texts from English Brown Corpus [14], containing 360,000 words with over 200,000 annotated [27]. SCWS is a dataset that contains 2003 word pairs along with their sentential contexts and human ratings on paired word similarities [16].

4.3 Methods

Several intrinsic tests have been applied to the embeddings. Given dataset SemCor [27], embeddings are generated and grouped according to their tagged senses. The average embeddings of a group is treated as the center of the group. Then the average of the average Normalized Euclidean Distance between every word pair in each group were measured, along with the average Normalized Euclidean Distance between every word pair of the centers. Also,

the average of the average cosine similarity score between every word-pair within each group and the average cosine similarity is calculated between every word pair of the center embeddings. This test is designed to measure the within-group distance and between-group distance of embedding models, relating to the distribution of embeddings depending on sense information learned by the model.

By using the dataset of SCWS [16], similarity evaluation is applied to the word embeddings where a Spearman rank-order correlation coefficient was calculated between cosine similarities of word embeddings and human ratings given by the dataset. The goal of this test to evaluate word embeddings capability of capturing sense relationships.

4.4 Algorithm based on PCA

Given embedding groups classified by their senses, we believe that with the large dimension size, not all dimensions are equally important in their word sense information representation. This PCA approach is inspired by the paper "All-but-the-Top: Simple and Effective Postprocessing for Word Representations" as an idea was brought up in the paper that the word embeddings are not zero-mean and they usually contain commmon components

which were extracted using PCA approaches [28]. As a result, we tried an algorithm based on PCA where for each sense group, the average mean is first calculated and subtracted from the sense group. Then the top d+2 PCA components are extracted with d being a set parameter. The algorithm finishes by adding the projected components to the average of the sense group and output the vector. The evaluation methods that we applied to the PCA algorithm is similar to the one that will be mentioned in the evaluation of our last algorithm. Basically, each sense group center is calculated by averaging post-processed vectors. Then the cosine similarity between pair-wise sense centers are compared with the path similarity of word senses provided by WordNet using Spearman's rank correlation test. The result of the correlation test is illustrated in Table 4.1. As shown in the table, although the correlation score increased by 3 ELMo models with a dimension size of 512, there failed to show a general performance improvement in all the models. Thus, we decided to take a more experimental approach by having the machine to try different combinations of dimensions in testing which ones have a more crucial role in the sense groups.

Algorithm 1 PCA for extracting common elements in sense groups

Input Given Sense Group S, embeddings $v(w)$,$\{w, w \in \text{S}\}$, and threshold parameter d.
calculate the mean vector.
$\mu \leftarrow \frac{1}{|S|}\sum_{w \in S} v(w)$, $\tilde{v}(w) \leftarrow v(w) - \mu$
Step 1: Compute the top d+1 PCA components
Step 2: Get the common embedding
$v'(w) \leftarrow \mu + \sum_{i=1}^{d}(u_i^T \tilde{v}(w) u_i)$
Output $v'(w)$

Model	Dim	Out	ρ_{PCA}	ρ_{mean}	Δ_ρ
BERT	768	-2	0.56530	0.591308	-0.026008
BERT	1024	-2	0.40665	0.475711	-0.069061
ELMo	256	0	0.55436	0.608066	-0.053706
ELMo	256	1	0.64007	0.633749	0.006321
ELMo	256	2	0.68201	0.652321	0.029689
ELMo	256	av	0.64918	0.635427	0.013753
ELMo	512	0	0.59265	0.644203	-0.051553
ELMo	512	1	0.73306	0.597286	**0.135774**
ELMo	512	2	0.74948	0.671543	**0.077937**
ELMo	512	av	0.72212	0.64714	**0.07498**
ELMo	1024	0	0.60177	0.672977	-0.071207
ELMo	1024	1	0.62365	0.686678	-0.063028
ELMo	1024	2	0.69659	0.721001	-0.024411
ELMo	1024	av	0.61454	0.699179	-0.084639

Table 4.1: Spearman's Correlation Coefficient for PCA algorithm: Dim is the embedding dimension; Out is the output layer; ρ_{PCA} is the correlation coefficient for embedding sense group centers after PCA algorithm is applied with the sense relationships; ρ_{mean} is the correlation coefficient for the original embedding sense group centers with the sense relationships; Δ_ρ is the difference between ρ_{PCA} and ρ_{mean} ($\Delta_\rho = \rho_{PCA} - \rho_{mean}$).

4.4.1 Our Algorithm

Based on the assumption that with a large dimension size as 1000, not every embedding dimension plays a role in contributing to each sense group. Here we propose a new method for determining the importance of specific dimension in determining senses in word embeddings by introducing a new algorithm for learning the significance of each dimension in word embeddings. Given the sense groups obtained from the SemCor dataset, the objective function in this algorithm is to maximize the average cosine similarity between all the word pairs in each sense group. At first, a weight matrix with the dimension of word embedding is initialized for each sense. Each dimension of this weight matrix represents the importance of a specific dimension to the sense. Later on, the weight matrices are put into training by the objective function defined.

During the training procedures, a mask matrix is generated with parameter N dimensions to be masked. Then the mask matrix is applied to the weight matrix, and the average cosine similarity of each group gets calculated. The gradient of the algorithm is defined to be the difference between the current similarity score and the previous similarity score times the masked dimensions. The weight matrix is updated during training with the gradient and learning rate.

The generation of the mask matrix involves two phases. In the first phase, the algorithm randomly generates N positions to be masked to ensure enough dimensions have been covered. After a certain number of epochs, the training enters a second phase where an exploration-exploitation policy is employed. The policy states that there is a chance of for the N numbers to be randomly generated and for the rest of the 1 - α chance, N numbers are randomly generated based on a weight which is the negated and normalized weight matrix. This policy results in the situation that the dimensions with higher wight have a smaller chance of getting masked. Also, l_1 regularization is applied to the gradient to encourage feature selection, and Adagrad algorithm is used to advance convergence. Pseudo-code for the proposed algorithm is demonstrated below, where n is the determined number of epochs for exploration, λ the parameter for l_1 regularization and ϵ a small number to prevent zero denominator in Adagrad.

After the weights are learned for word embedding dimensions, we pick out the ones with low importance and test if the rest of the dimensions are enough to represent the sense group. Therefore, the remaining dimensions of the word embedding should be the mere parts in the group of word embeddings for representing that certain word sense. We propose to use the mean of

the vectors with remaining dimensions of the same sense group as the sense vector of that group.

Algorithm 2 algorithm for learning dimension weights

for each sense group SG **do**
  initialize weights w, learning rate γ_0, Adagrad weights matrix gti
  initialize $S_{pre} \leftarrow \sum_{v_i, v_j \in SG, i \neq j} Cosine(v_i, v_j)$
  for each epoch i **do**
    if $i < n$ **then**
      randomly generate N
      $N_1, \cdots, N_N$
    else
      generate N numbers according to explore-exploitation policy
      $N_1, \cdots, N_N$
    end if
    mask arrays on dimensions $N_1, \cdots, N_N$
    $S_{cur} \leftarrow \sum_{v_i, v_j \in SG, i \neq j} Cosine(v_i, v_j)$
    $grad = (S_{pre} - S_{cur}) * (w - 1) - \lambda * sign(w)$
    $gti \mathrel{+}= grad^2$
    $w \leftarrow \dfrac{w + grad * \gamma_i}{\epsilon + \sqrt{gti}}$
  end for
end for

Chapter 5

Test and Evaluation

5.1 Test Results

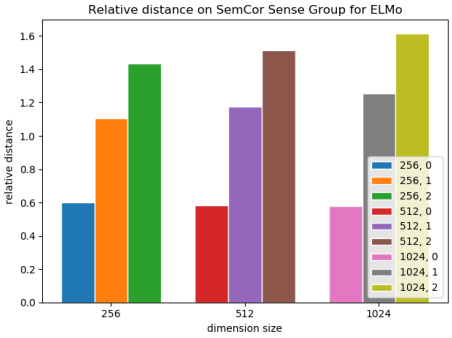
After the normalized Euclidean distances and cosine similarities were evaluated on word embedding groups sorted by word senses of the SemCor corpus, a relative distance was calculated by dividing the average within-group Euclidean distance by the average between-group Euclidean distance and a relative similarity by dividing the average within-group Cosine similarity score by the average between-group cosine similarity score. The results for the relative distances of ELMo models are shown in figure 5.1a and it can be noticed that the closer the output layer is to the input layer, the smaller relative distance the embedding groups have. Similarly, figure 5.1b presents the relative similarity score of Elmo models, where the closer the output layer is to the input layer, the larger similarity score the embedding groups

have. Figure 5.1c shows the Spearman Rank-Order Correlation Coefficient ρ for the SCWS dataset. The result adheres to the last result that the layer closer to the input layer has a better performance, except that ELMo model (1024) has a slightly small coefficient in the first layer than the second with a difference less than 0.0005. In addition, the performance is better in this test for embeddings with an increasing dimension size.

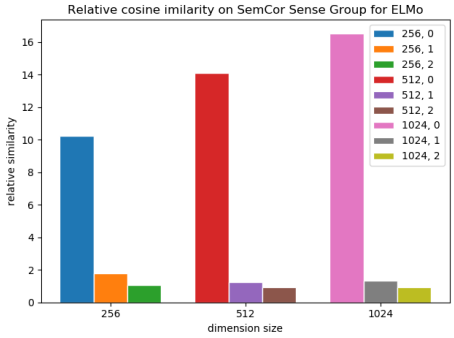
Figure 5.2 demonstrates the relative Euclidean distance and relative similarity on SemCor corpus with the correlation test on SCWS dataset results for the two BERT models. The BERT models displayed an interesting pattern where they have a higher within-group distance than between-group distance with a relative distance value bigger than 1 and a lower within-group similarity than between-group cosine similarity with a relative similarity value smaller than 1.

Table 5.1 contains all the test results for Euclidean distances and cosine similarities on SemCor corpus, and Spearman correlation coefficients on SCWS dataset. Despite the larger within-group distance than between-group distance, both BERT models still outperformed the FLAIR models in the coefficient test and BERT model with a dimension of 768 ranks 6 in the coefficient test. One possible explanation is that with the attention mechanism of Bert, as

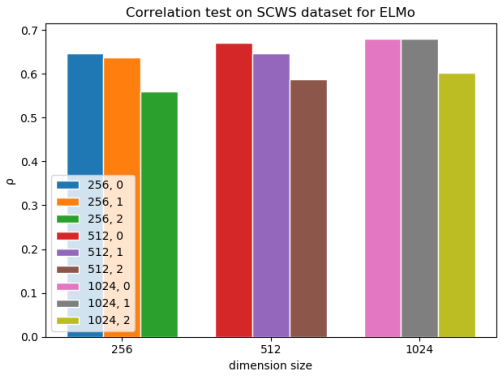
each word embedding is highly contextualized, there is a higher chance of learning sense relation information than sense discriminating information. Also, contrary to a performance increase in the other two models with rise of dimension size in the coefficient task, BERT resulted in a decrease of the correlation coefficient. However, since only two models of BERT are tested here, more tests need to be done to come to an see if increasing embedding dimension results in increasing noises in the BERT embedding.



(a) Relative average distances on SemCor Sense Groups for ELMo models. Relative average distance is calculated by average within-group Euclidean distance divided by average between-group Euclidean distance.



(b) Relative cosine similarity scores on SemCor Sense Groups for ELMo models. Relative cosine similarity is calculated by average within-group similarity score divided by average between-group similarity score.



(c) Correlation test for correlation coefficient ρ on SCWS dataset for ELMo models

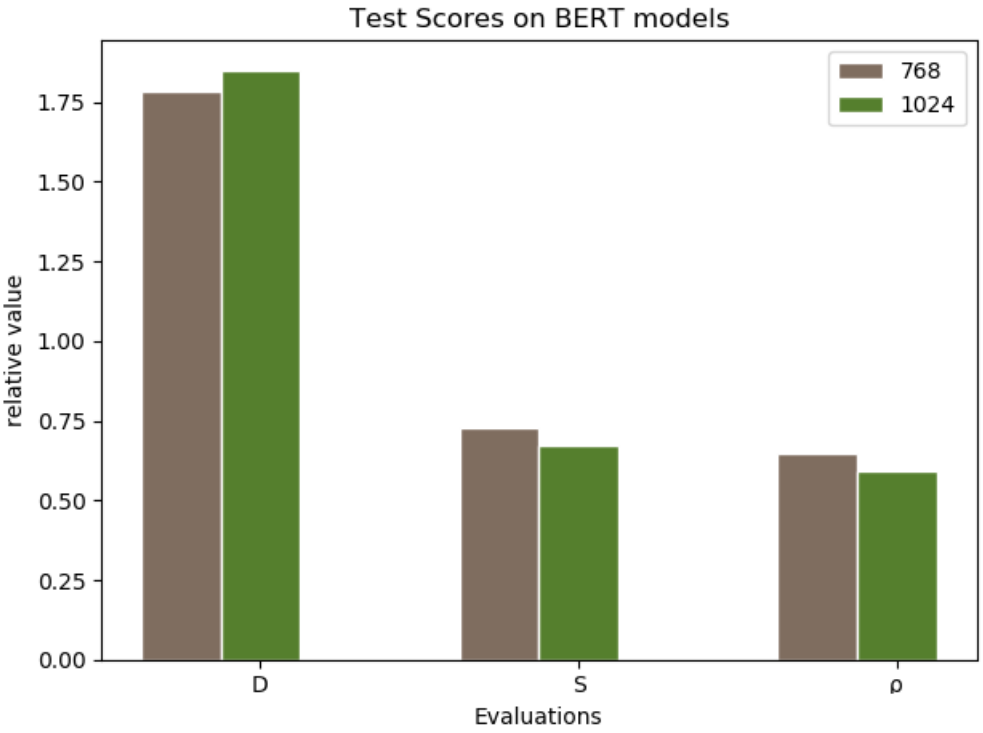


Figure 5.2: Relative average distance, relative average similarity on SemCor Sense Groups and correlation test for correlation coefficient ρ on SCWS dataset

Model	Dim	Out	Ave Dis	C Dis	Ave Cos	C Cos	ρ_{SCWS}
BERT	768	-2	34.3856	19.30123	0.497062	0.683877	0.647989
BERT	1024	-2	39.74155	21.47615	0.58236	0.864297	0.594294
ELMo	256	0	10.20746	17.06137	0.596261	0.058296	0.647328
ELMo	256	1	17.01257	15.38437	0.511956	0.283216	0.63676
ELMo	256	2	18.35877	12.79827	0.451013	0.429331	0.560226
ELMo	256	av	17.06634	14.2695	0.493568	0.341095	0.62173
ELMo	512	0	13.58388	23.34382	0.595626	0.0423	0.670698
ELMo	512	1	24.3524	20.71935	0.541471	0.439504	0.646561
ELMo	512	2	26.29227	17.36125	0.440618	0.463392	0.586672
ELMo	512	av	24.49851	19.42738	0.493186	0.403634	0.650462
ELMo	1024	0	18.40521	31.8018	0.592719	0.035835	0.679799
ELMo	1024	1	34.63281	27.59308	0.478008	0.356505	0.680234
ELMo	1024	2	37.62374	23.28052	0.39284	0.421985	0.601395
ELMo	1024	av	34.65655	26.45345	0.454141	0.347544	0.67123
Flair	2048	-1	49.67542	38.48288	0.559736	0.508835	0.555379
Flair	4096	-1	71.76762	52.52496	0.433499	0.332308	0.498259

Table 5.1: Test results with model, embedding dimension size (Dim), output layer (Out), average of group average pair-wise Normalized Euclidean Distance (Ave Dis), average of pair-wise center vector Euclidean Distance (C Dis), Average of group average pair-wise cosine similarity (Ave Cos), average of pair-wise center vector cosine similarity (C Cos) and Spearman's coefficient on SCWS dataset (ρ_{SCWS})

5.2 Evaluation of Algorithm

Figure 5.3 and figure 5.4 are examples generated by visualizing the dimension weights for sense "obtain.v.01", which has a definition of "come into possession

of", and sense "have.v.01", which has a meaning of "have or possess, whether in concrete or abstract way", after applying our algorithm on the ELMo model with a dimension of 512, using the second layer as the output layer. As shown in the graph, the value of dimensions varies with some dropped to 0. The values of dimension weights can be interpreted as the value of that certain dimension in representing the current sense "obtain.v.01". The red line in the graph is the threshold where every dimension with weights below the threshold will be masked out, which is set to 0.6 here. Figure 5.3 and figure 5.6 show the mean word embedding of sense group "obtain.v.01" and "have.v.01" with orange dots representing the dimensions that have been masked out to 0.

Figure 5.7 and figure 5.7 are graphs based on the ELMo embedding model with a dimension size of 512 and the third network layer as the output layer. Figure 5.7 is a graph of 20 selected sense groups with 100 embeddings each. Figure 5.8 is a graph of 20 selected sense groups with 100 embeddings each and the dimensions of the embedddings with a weight value smaller than 0.5 is masked to 0. Both of the two figures are being projected from 512 dimensions to 2 dimensions using Linear Discriminant Analysis. It can be seen that taken into account of the scale size, the clusters generated after

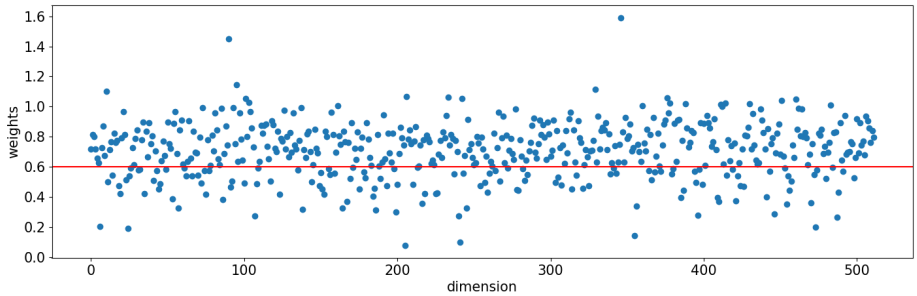


Figure 5.3: Dimension weights visualization for sense group“obtain.v.01” in ELMo embeddings (512 dimensions and second output layer)

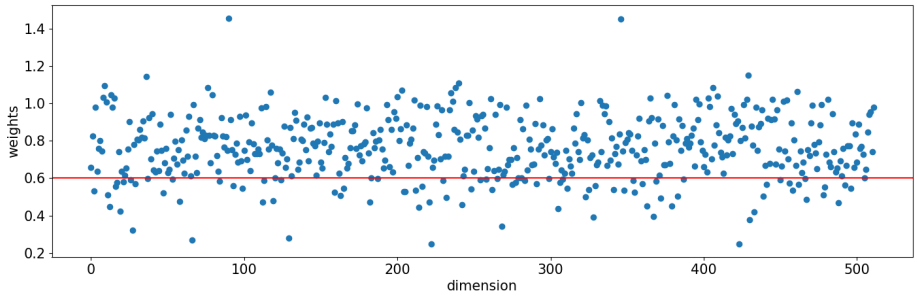


Figure 5.4: Dimension weights visualization for sense group “have.v.01” in ELMo embeddings (512 dimensions and second output layer)

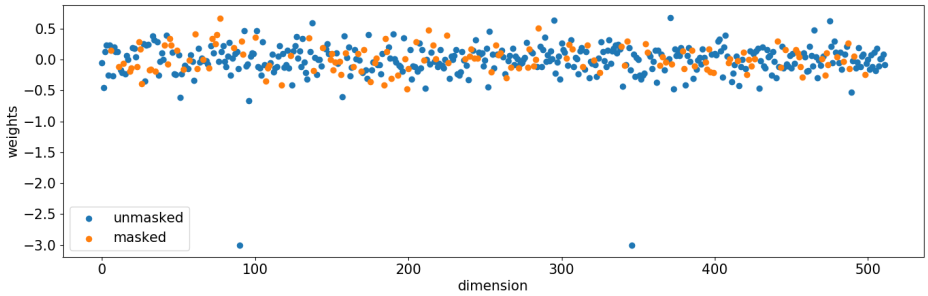


Figure 5.5: mean embedding of sense group "obtain.v.01" in ELMo embeddings (512 dimensions and second output layer) with orange dots representing the dimension values masked out to 0

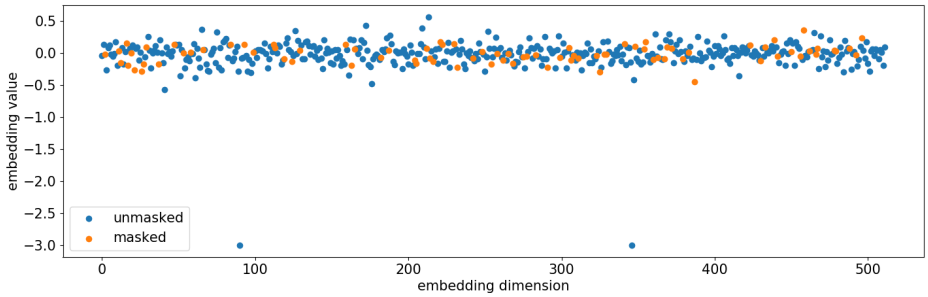


Figure 5.6: mean embedding of sense group "have.v.01" in ELMo embeddings (512 dimensions and second output layer) with orange dots representing the dimension values masked out to 0

dimensions are masked out are of a smaller distance than the original ones. Although the original embeddings show a large distance between three main cluster of clusters, the processed embeddings demonstrated a better separation of cluster groups when the original one generates a cluster group where all clusters are interlaced with each other.

To evaluate the quality of weights learned by our algorithm, we masked out dimensions of embeddings with relatively low weight value in sense groups and performed the following evaluations to compare with the unmasked embeddings: pair-wise cosine similarity within each sense group and the Spearman's Rank-Order Correlation Coefficient ρ between the cosine similarity of sense vectors extracted and path similarity scores provided by WordNet. The path similarity score is based on the shortest path that connects the senses based on a hypernym/hyponym relationship [11]. The mean embeddings of each sense group are extracted as the sense embeddings and a number of thresholds for weights were tested to find an optimum threshold for masking dimensions. Table 5.2 contains the results of embedding weights trained after 1000 epochs with the average number of dimensions that get masked out in sense groups and evaluation scores for unmasked and masked embeddings in correlation test and cosine similarity test. As can be seen in the table 5.2,

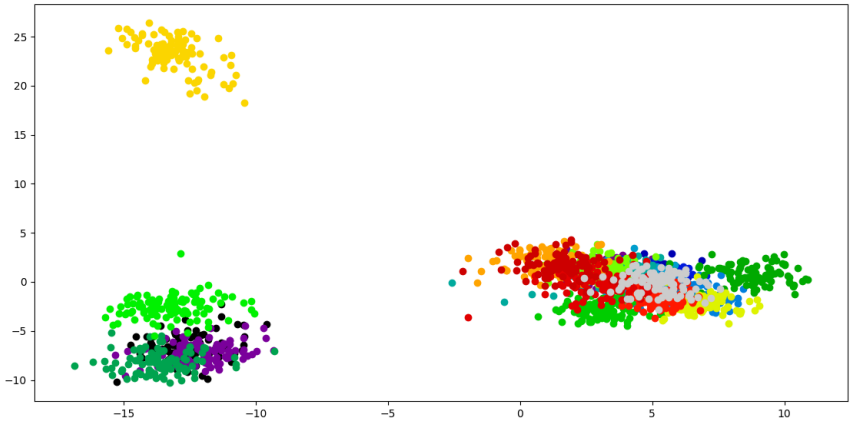


Figure 5.7: Graph of 20 selected sense groups with 100 embeddings each for embedding model of ELMo and a dimension size of 512 (third output layer). The projection of dimensions from 512 to 2 is done by Linear Discriminant Analysis.

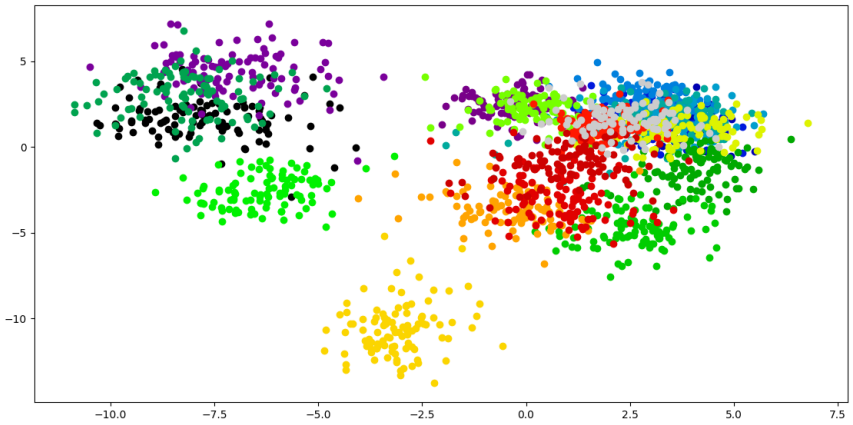


Figure 5.8: Graph of 20 selected sense groups with 100 embeddings each for embedding model of ELMo and a dimension size of 512 (third output layer) with dimensions with a weight smaller than 0.5 being masked to 0. The projection of dimensions from 512 to 2 is done by Linear Discriminant Analysis.

after masking out the dimensions, the cosine similarity and sense vector correlation did not vary much, and some of the scores increase, which proves that the embedding dimensions that are masked out do not contribute to the representation of the sense groups.

From table 5.2, it can be noticed that for certain models such as ELMo (second output layer) and Flair model, with the insignificant embedding dimensions masked out, the model sense group shows a better ability for representing the sense relationships. For instance, the correlation coefficient ρ increased from 0.266 to 0.369 in ELMo model with 256 dimension size (average embedding) after an average number of 199 dimensions get masked to 0 throughout the sense groups. Even if the dimension mask does not increase the word sense correlations, the correlation performances were not significantly influenced. This result proves our assumption that there are some dimensions in the embeddings that are negligible in representing senses, which applies to all word embedding models and verifies that our algorithm does learn the importance of different dimensions in sense groups. Overall, we can also see that the average cosine similarity within sense groups increase significantly after the dimensions are masked out for all the embedding models, which in addition proves that our algorithm is guided to learn the dimension

weights based on the objective function given. Taking a more detailed look at each embedding models, we can conclude that in both BERT models, over 100 dimensions can be masked out without substantially influencing the performance of both tasks. For the ELMo models, the number of embeddings that we can discard increases with the dimension size and with the distance of the output layer to the input layer. One other thing to notice is that the number of dimensions able to be masked out in the second output layer for ELMo models is approximate over half of the embedding size. This result also corresponds to our result from the preliminary test where ELMo model output layers performances were ranked $1 > 2 > 3$, while in this test, the dimension numbers to be masked out are ranked as follows: $3 > 2 > 1$. The more dimensions which can be masked out means that more insignificant dimensions exit in layer 3 than layer 2, thus creating more noise in the tests in the last section. In general, BERT models have less embedding dimensions to be masked out than ELMo and Flair models.

Another pattern that we notice in the experiment results is that the verb sense groups tend to have less number of dimensions getting masked out. Less trivial dimensions could be caused by the fact that verb sense groups have more possible forms of words belonging to the same group. Table 5.3 contains

Model	Dim	Out	Dim_{masked}	$\rho_{unmasked}$	ρ_{masked}	$cos_{unmasked}$	cos_{masked}
BERT	768	-2	125	0.26814	0.26286	0.4971	0.5187
BERT	1024	-2	146	0.27423	0.26575	0.5811	0.5983
ELMo	256	0	95	0.12016	**0.16017**	0.5959	0.6134
ELMo	256	1	105	0.30903	**0.37377**	0.5119	0.5787
ELMo	256	2	218	0.2852	0.3042	0.4507	0.6914
ELMo	256	av	199	0.26553	**0.36945**	0.4932	0.6729
ELMo	512	0	136	0.17058	0.17336	0.5957	0.6051
ELMo	512	1	181	0.27967	0.25318	0.5414	0.5908
ELMo	512	2	281	0.29577	**0.36943**	0.4404	0.5346
ELMo	512	av	207	0.2949	0.30047	0.4930	0.5470
ELMo	1024	0	179	0.18504	0.17263	0.5930	0.5945
ELMo	1024	1	198	0.30897	0.30175	0.4783	0.4971
ELMo	1024	2	608	0.28406	**0.30675**	0.3927	0.4915
ELMo	1024	av	406	0.28331	**0.27204**	0.4542	0.5086
Flair	2048	-1	670	0.24891	0.28516	0.5560	0.6084

Table 5.2: Cosine similarity and correlation test results for unmasked and masked word embeddings: embedding model (Model), dimension size (Dim), output layer (Out), the average number of dimensions that are masked to zero in embedding sense groups (Dim_{masked}), the correlation coefficient of original embedding sense group centers and sense relations ($\rho_{unmasked}$), the correlation coefficient of embedding sense group centers with dimensions masked to 0 (ρ_{masked}), the average within-group cosine similarity for the original embeddings ($cos_{unmasked}$) and the average within-group cosine similarity after the dimensions are masked out ($cos_{unmasked}$).

Model	Dim	Out	Sense	N_{masked}	Sense	N_{masked}	Sense	N_{masked}
BERT	768	-2	ask.v.01	75	three.s.01	208	man.n.01	58
BERT	1024	-2	ask.v.01	40	three.s.01	211	man.n.01	116
ELMo	256	0	ask.v.01	78	three.s.01	182	man.n.01	242
ELMo	256	1	ask.v.01	78	three.s.01	99	man.n.01	212
ELMo	256	2	ask.v.01	241	three.s.01	243	man.n.01	212
ELMo	256	av	ask.v.01	155	three.s.01	120	man.n.01	227
ELMo	512	0	ask.v.01	103	three.s.01	28	man.n.01	295
ELMo	512	1	ask.v.01	144	three.s.01	105	man.n.01	156
ELMo	512	2	ask.v.01	334	three.s.01	300	man.n.01	311
ELMo	512	av	ask.v.01	205	three.s.01	122	man.n.01	317
ELMo	1024	0	ask.v.01	174	three.s.01	44	man.n.01	331
ELMo	1024	1	ask.v.01	60	three.s.01	106	man.n.01	220
ELMo	1024	2	ask.v.01	568	three.s.01	827	man.n.01	708
ELMo	1024	av	ask.v.01	181	three.s.01	351	man.n.01	426
Flair	2048	-1	ask.v.01	193	three.s.01	862	man.n.01	1883

Table 5.3: embedding models with specific word embedding sense groups (Sense) and the embedding dimension numbers masked out in according groups (N_{masked}): “ask.v.01” is a verb word sense with a meaning of “inquire about”; “three.s.01” is an adjective word sense with a meaning of “being one more than two”; “man.n.01” is a noun word sense with a meaning of “an adult person who is male (as opposed to a woman)” [11].

the number of dimensions that get discarded with some of the common sense groups. The sense ”ask.v.01” in the table has a definition of ”inquire about” and have some possible forms including ”ask”, ”inquire” and ”enquire”.

Chapter 6

Conclusion

This paper demonstrates a novel approach to word embedding interpretation. Mainly focused on context-based word embeddings ability to distinguish and learn relationships in word senses, we first approached the subjective by conducting analysis on the pre-trained word embeddings in their distribution given sense groups. The second part of this study proposes an algorithm for learning and visualizing the importance of dimension weights in sense groups. After training the weights for word dimensions, the dimensions with a less importance were masked out and tested using two evaluations. A conclusion can be drawn from the results that there are some dimensions that do not contribute to the representation of sense groups and our algorithm can distinguish the importance of dimensions.

6.1 Limitation and Future Work

There are several limitations to this work. First, for the evaluation of the sense vectors, the path similarity provided by the WordNet [11] may not be the best to fit human judgements. Second, the current tests were limited by the dataset corpus we used. For future works, the algorithm can be extended to the extraction of sense vectors although currently the number of sense vectors generated are constrained by the size of the annotated corpus. Also, the applications of the algorithm can theoretically be applied to other grouped embeddings where words may be grouped by topics and concepts, which would require more explorations and fine tunings from the current model.

Bibliography

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/C18-1139`.

[2] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *CoRR*, abs/1601.03764, 2016. URL `http://arxiv.org/abs/1601.03764`.

[3] Amir Bakarov. A survey of word embeddings evaluation methods. *CoRR*, abs/1801.09536, 2018. URL `http://arxiv.org/abs/1801.09536`.

[4] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress

in statistical language modeling. *CoRR*, abs/1312.3005, 2013. URL http://arxiv.org/abs/1312.3005.

[5] Billy Chiu, Anna Korhonen, and Sampo Pyysalo. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 1–6. Association for Computational Linguistics, 2016. doi: 10.18653/v1/W16-2501. URL http://aclweb.org/anthology/W16-2501.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[7] Allyson Ettinger and Tal Linzen. Evaluating vector space models using human semantic priming results. pages 72–77, 01 2016. doi: 10.18653/v1/W16-2513.

[8] Allyson Ettinger, Philip Resnik, and Marine Carpuat. Retrofitting sense-specific word vectors using parallel text. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1378–

1383, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1163. URL http://www.aclweb.org/anthology/N16-1163.

[9] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. *CoRR*, abs/1411.4166, 2014. URL http://arxiv.org/abs/1411.4166.

[10] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. Sparse overcomplete word vector representations. *CoRR*, abs/1506.02004, 2015. URL http://arxiv.org/abs/1506.02004.

[11] C. Fellbaum and G. Miller. *Building Semantic Concordances*. MITP, 1998. ISBN 9780262272551. URL https://ieeexplore.ieee.org/document/6287676.

[12] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 406–414, New York, NY, USA,

2001. ACM. ISBN 1-58113-348-0. doi: 10.1145/371920.372094. URL http://doi.acm.org/10.1145/371920.372094.

[13] J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

[14] W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979. URL http://icame.uib.no/brown/bcm.html.

[15] Daniela Gerz, Ivan Vulic, Felix Hill, Roi Reichart, and Anna Korhonen. Simverb-3500: A large-scale evaluation set of verb similarity. *CoRR*, abs/1608.00869, 2016. URL http://arxiv.org/abs/1608.00869.

[16] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012.

[17] Kyoung-Rok Jang and Sung-Hyon Myaeng. Elucidating conceptual properties from word embeddings. In *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, pages 91–95. Association for Computational Linguistics, 2017. doi: 10.18653/v1/W17-1911. URL http://aclweb.org/anthology/W17-1911.

[18] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016. URL `http://arxiv.org/abs/1607.01759`.

[19] David A. Jurgens, Peter D. Turney, Saif M. Mohammad, and Keith J. Holyoak. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 356–364, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=2387636.2387693`.

[20] Aykut Koç, Ihsan Utlu, Lutfi Kerem Senel, and Haldun M. Özaktas. Imparting interpretability to word embeddings. *CoRR*, abs/1807.07279, 2018. URL `http://arxiv.org/abs/1807.07279`.

[21] Thomas K. Landauer. A solution to plato ' s problem : The latent semantic analysis theory of acquisition , induction , and representation of knowledge. 1997.

[22] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[23] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2418–2424. AAAI Press, 2015. ISBN 0-262-51129-0. URL `http://dl.acm.org/citation.cfm?id=2886521.2886657`.

[24] Hongyin Luo, Zhiyuan Liu, Huan-Bo Luan, and Maosong Sun. Online learning of interpretable word embeddings. In *EMNLP*, 2015.

[25] Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. The role of context types and dimensionality in learning word embeddings. *CoRR*, abs/1601.00893, 2016. URL `http://arxiv.org/abs/1601.00893`.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013. URL `http://arxiv.org/abs/1310.4546`.

[27] George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and

Robert G. Thomas. Using a semantic concordance for sense identification. In *HLT*, 1994.

[28] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. *CoRR*, abs/1702.01417, 2017. URL `http://arxiv.org/abs/1702.01417`.

[29] Brian Murphy, Partha Talukdar, and Tom Mitchell. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING 2012*, pages 1933–1950. The COLING 2012 Organizing Committee, 2012. URL `http://aclweb.org/anthology/C12-1118`.

[30] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250, December 2012. ISSN 0004-3702. doi: 10.1016/j.artint.2012.07.001. URL `http://dx.doi.org/10.1016/j.artint.2012.07.001`.

[31] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings

per word in vector space. *CoRR*, abs/1504.06654, 2015. URL http://arxiv.org/abs/1504.06654.

[32] Dai Quoc Nguyen, Dat Quoc Nguyen, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. A mixture model for learning multi-sense word embeddings. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 121–127, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-1015. URL http://www.aclweb.org/anthology/S17-1015.

[33] Sungjoon Park, JinYeong Bak, and Alice Oh. Rotated word vector representations and their interpretability. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 401–411, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1041. URL http://www.aclweb.org/anthology/D17-1041.

[34] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural*

Language Processing (EMNLP), pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[35] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *CoRR*, abs/1705.00108, 2017. URL http://arxiv.org/abs/1705.00108.

[36] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. URL http://arxiv.org/abs/1802.05365.

[37] Mohammad Taher Pilehvar and Nigel Collier. De-conflated semantic representations. *CoRR*, abs/1608.01961, 2016. URL http://arxiv.org/abs/1608.01961.

[38] Mohammad Taher Pilehvar and Roberto Navigli. From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. *Artif. Intell.*, 228:95–128, 2015.

[39] Joseph Reisinger and Raymond J. Mooney. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010*

Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 109–117. Association for Computational Linguistics, 2010. URL http://aclweb.org/anthology/N10-1013.

[40] Anna Rogers, Shashwath Hosur Ananthakrishna, and Anna Rumshisky. What's in your embedding, and how it predicts task performance. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2690–2703, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C18-1228.

[41] Sascha Rothe and Hinrich Schütze. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1173. URL http://www.aclweb.org/anthology/P15-1173.

[42] Hinrich Schütze. Automatic word sense discrimination. *Comput.*

Linguist., 24(1):97–123, March 1998. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972719.972724.

[43] Jamin Shin, Andrea Madotto, and Pascale Fung. Interpreting word embeddings with eigenvector analysis. 2018.

[44] Anders Søgaard. Evaluating word embeddings with fmri and eye-tracking. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 116–121. Association for Computational Linguistics, 2016. doi: 10.18653/v1/W16-2521. URL http://aclweb.org/anthology/W16-2521.

[45] Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard H. Hovy. SPINE: sparse interpretable neural embeddings. *CoRR*, abs/1711.08792, 2017. URL http://arxiv.org/abs/1711.08792.

[46] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Association for Computational Linguistics, 2015. doi: 10.18653/v1/D15-1174. URL http://aclweb.org/anthology/D15-1174.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[48] Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. Hyperlex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835, 2017.

[49] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-5446.

[50] Xuefeng Yang and Kezhi Mao. Learning multi-prototype word embedding from single-prototype word embedding with integrated knowledge. *Expert*

Syst. Appl., 56(C):291–299, September 2016. ISSN 0957-4174. doi: 10.1016/j.eswa.2016.03.013. URL http://dx.doi.org/10.1016/j.eswa.2016.03.013.

[51] Zi Yin and Yuanyuan Shen. On the dimensionality of word embedding. *CoRR*, abs/1812.04224, 2018. URL http://arxiv.org/abs/1812.04224.

[52] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*, 2015.

[53] Alexey Zobnin. Rotations and interpretability of word embeddings: the case of the russian language. *CoRR*, abs/1707.04662, 2017. URL http://arxiv.org/abs/1707.04662.