Tweaking Word Vectors for Domain-specific Applications

Yingxin Zhang

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2018

Abstract

Word embeddings has received significant amount of attention in mainstream Natural Language Processing (NLP). They are usually used as features to boost performance on a variety of downstream NLP tasks. Many existing word embeddings have been released by different research organizations. However, such pre-trained word embeddings are trained on generic corpus, which limits the direct use of them in domain-specific tasks. In this paper, we present several approaches of using pre-trained word embeddings to guide word embedding learning on domain-specific corpus. We evaluated our approaches on two domain-specific dataset: Quora questions dataset and Pubmed RCTs dataset. Our evaluation results in downstream classification tasks demonstrated the effectiveness of our approaches.

Acknowledgements

Firstly I would like to thank my supervisors Adam Lopez, Alan Nichol and Joey Faulkner for their advice and patience throughout the project. The numerous discussions and meetings made this project a real joy to work on. Furthermore, I would also like to thank my friends and boyfriend for their constant encouragement, and my family for financial support during my MSc studies.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Yingxin Zhang*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Word embedding is a recent popular technique in Natural Language Processing (NLP) area. They are real-valued vectors that map words into a continuous vector space, and capture a range of interesting semantic relations among lexical items. Words that have similar meaning are mapped to close vectors in the vector space.

Many downstream NLP tasks take word embeddings as input features for better performance. However, it is troublesome to prepare word embeddings because current word embedding methods need very huge corpus of texts for generating exact word vectors. A primary reason for the increasing use of word embeddings in mainstream NLP is some recently released pre-trained word embeddings from different groups. For example, Google has released a pre-trained Word2Vec model containing 300-dimensional vectors for 3 million words [1], and Stanford NLP Group researchers released Glove (Global Vectors for Word Representation) model for distributed word representation [2]. They are currently among the most accurate word embeddings. These pre-trained word embeddings are usually learned on massive text corpus which contains billions of words.

Many developers prefer to use pre-trained word embeddings in their downstream applications to shorten their training time, or for a better quality word vector model when training datasets are small. However, such pre-trained word embeddings are usually produced on generic corpus, which may not fit the data well. If a downstream NLP task involves a particular domain, it may cause the following problems:

[1] https://code.google.com/archive/p/word2vec/

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

- No word embeddings for domain specific words, because they are not present in generic corpus which pre-trained word embeddings trained on. These words are Out of Vocabulary (OOV) words in pre-trained models.
- Even if some domain specific words do have embeddings, they may fail to capture in-domain sense or domain specific relationship due to semantic variation. For example, the word 'bat' is a kind of mammals in biology domain, whereas it also refers to cricket player in sports domain.

Therefore, if we want to apply word embeddings in a downstream task related to a specific domain, it is better to train a new word vector model on domain-specific corpus than using pre-trained embeddings. A domain specific corpus is a set of texts related to a particular domain. The usage of words in domain specific corpus are limited to texts and contexts that have special meanings to that domain. However, in most cases, the in-domain texts are sparse that prevent words from converging to informative vectors.

1.2 Objective

Our work is concerned with the approach of producing better word embeddings from an in-domain corpus based on existing pre-trained word embedding sets, in order to gain performance in domain-specific downstream tasks. For example, assume that we want to build a classification model on medical documents but have limited text collections in this domain. Meanwhile, we may have a set of pre-trained word embeddings, who is trained in news or tweets text corpus. Their type of texts is not necessarily limited to the medical field. Can this knowledge from multiple domains be useful to help us learn better word embeddings in the medical domain, such that the domain-specific application can benefit from the addition of information? In this paper, we address this question: given a set of pre-trained word embeddings and domain-specific corpus, we utilize the pre-trained word embeddings to guide word vector modeling on the domain-specific corpus, such that the tuned model can yield better word embedding sets than pre-trained one to gain performance in downstream tasks.

1.3 Contributions

The contributions of this paper can be summarized as follows:

- Based on the basic Skip-Gram model, a regularization-based Skip-Gram model for domain adaptation on word embedding level is developed, and other approaches for adapting word embedding are tested as well.
- We show that our approaches can yield better domain word embeddings that boost performance on downstream classification tasks.
- We provide several guidelines for obtaining good word embeddings for domain-specific tasks based on our experiments results on specific tasks.

1.4 Overview

The paper is organized as follows: In Section 2 we provide some background material. In Section 3 we describe the domains we work on. In Section 4 we explain the pre-trained word vector model and present all kinds of approaches we used for domain word embeddings learning. In Section 5 we explain the design of evaluation tasks, the experiments setup, their results, and analysis of the results. In Section 6 we review our work in its totality, draw overall conclusions and sketch future plan.

Chapter 2

Background

2.1 Natural language processing

Natural language processing (NLP) is an important branch in the fields of computer science and artificial intelligence, which particularly concentrates on teaching computers to process, analyze and generate human (natural) language data. Formerly, many successful natural language processing systems relied on complex hand-written rules. However, with the introduction of machine learning in recent years, more and more NLP systems applied machine learning algorithms and achieve pronounced improvements. Machine learning algorithms usually take as input a set of features that represent input data and automatically learning the rules. Such models are robust to unfamiliar inputs and can be more accurate by simply augment more training data.

2.2 Classification task in NLP

Classification task is a typical task in machine learning area, which trains models to map elements drawn from specific samples space to a set of categories. It can be described as predicting label y given input features x, by modeling $p(y|x)$ for probability distribution over all possible labels. The classification paradigm is applied widely in many NLP tasks due to its flexibility and simplicity. It is a task of assigning predefined label to text document. The labels can be topics of stories, tags of products, sentiments, etc. The number of categories can be binary or multiple. For example, in a multi-class classification task, the inputs can be short sentences for the abstract of a paper, and outputs are labels of those sentences indicating their semantic roles such as 'Background', 'Method', 'Result', etc.

2.3 An overview of word vectors

2.3.1 Traditional word vector model

Traditionally, word has been represented with word vector model in a naive way that each unique term is considered as an independent dimension of high-dimensional discrete space (Salton and Buckley, 1988). Within this framework, word vectors have same dimensionality with vocabulary size and only one dimension is on, which is so-called 'one-hot' encoding. For example, we have only three words in vocabulary: 'red', 'yellow' and 'green'. We create a separate dimension for each possible word, and put a 1 in the column which represents a certain word and 0 in other columns. Hence the one-hot vector for word 'red' is $[1,0,0]$, 'yellow' is $[0,1,0]$ and 'green' is $[0,0,1]$.

This word vector model has some drawbacks: One-hot vectors are high-dimensional that vector size grows with vocabulary size. Downstream tasks may suffer from *the curse of dimensionality* thus becomes less computationally efficient. Each dimension value represents a unique term in a discrete space, usually fail to capture semantic or syntactic relationship between words. For example, the word *lemon* may be represented as vector $[0,1,0,...,0]$ and the word *vanilla* may be represented as vector $[0,0,1,...,0]$. According to the one-hot vector space, *lemon* and *vanilla* are orthogonal so they follow that $\overrightarrow{lemon} \cdot \overrightarrow{vanilla} = 0$. This is as dissimilar as *lemon* and *python*. At test time, such encoding method cannot handle words out of vocabulary.

2.3.2 Distributed word representations

The limitations of aforementioned one-hot encoding motivated the use of word embeddings. Conceptually, word embedding involves a machine learning technique to map words from high-dimensional discrete space to real-valued vectors in low-dimensional continuous space. Each dimension of word embeddings represent a latent feature of word, ideally capturing syntactic and semantic properties among words.

Training word embeddings in a neural network architecture was originally introduced by (Bengio et al., 2003), who learned a distributed representation for words with the probability for word sequences in a neural language model simultaneously. The objective of this model is predicting next word in a sequence, and task-specific word embeddings are learned as part of the deep learning model. Since then, the learning of word embeddings has received considerable attention. They include Log-

Bilinear Language Model (LBL) proposed by (Mnih and Hinton, 2007), ivLBL model proposed by (Mnih and Kavukcuoglu, 2013), CW model proposed by (Collobert and Weston, 2008), CBOW and skip-gram model proposed by (Mikolov et al., 2013b), and skip-gram with negative sampling model proposed by (Mikolov et al., 2013a). Most of these methods are based on the distributional hypothesis: words that appear in similar contexts tend to have similar semantic meanings (Harris, 1954). So they are prediction-based approaches that given context words c and a target word w, they try to find word representations that allow model to predict w from c or predict c from w. Another group of word embedding models are Hellinger PCA proposed by (Lebret and Collobert, 2013), GloVe model proposed by (Pennington et al., 2014), TSCCA proposed by (Dhillon et al., 2012). They follow a reconstruction approach that learn word embeddings from original co-occurrence matrix.

Currently, the Word2Vec method (Mikolov et al., 2013b) gained much popularity in the field of learning word embeddings and brought the word embeddings to the mainstream in NLP space. It includes two models: a continuous bag-of-words model (CBOW) and a skip-gram model, both learn word embeddings from unlabelled data. Skip-gram models have been found to be highly efficient in learning word embeddings from huge amounts of unlabelled text data and detect various semantic and syntactic relationships. The mean idea behind Skip-Gram model is: it trains a single hidden layer neural network to predict the nearby word of a given word token from large corpora.

It is well to be reminded that Word2Vec model aims to produce task-independent word embeddings from unlabelled corpus that can be used as input features in many other downstream NLP tasks. We can call this kind of approach a form of transfer learning. (Mikolov et al., 2013a) claimed that word embeddings generated from neural network can encode good syntactic and semantic relations. A classic example is analogy questions in the form of 'A1 is to A2 as B1 is to B2' can be solved by using simple vector operation. For example: $\overrightarrow{king} - \overrightarrow{man} = \overrightarrow{queen} - \overrightarrow{woman}$, $\overrightarrow{cats} - \overrightarrow{cat} = \overrightarrow{mice} - \overrightarrow{mouse}$.

2.4 Related work

2.4.1 Ensembles of embedding sets

Some of existing works focus on combining several embedding sets for better quality. (Yin and Schütze, 2015) proposed an ensemble method of combining multiple

word embedding sets from different word vector algorithms to learn meta-embeddings and shows improvement on intrinsic tasks such as word similarity and analogy task. (Garten et al., 2015) also demonstrated that composing diverse word embeddings into hybird representations can effectively leverage strengths of individual embedding set. (Sarma et al., 2018) proposed an algorithm to obtain domain adapted embeddings by projecting generic word embeddings and domain-specific word embeddings via Canonical Correlation Analysis (CCA) and then linearly combining them using convex optimization.

2.4.2 Incorporating domain knowledge

There are other studies focusing on learning domain-specific word embeddings in certain domain by adding additional domain knowledge to existing word embeddings. (Patel et al., 2017) proposed a modification of CBOW algorithm to introduce medical lexicon information to generic word embeddings. (Roy et al., 2017) develop a comprehensive algorithm to incorporate diverse types of annotated cybersecurity domain knowledge in domain word embeddings.

2.4.3 Domain adaptation

Domain adaptation is a field related to machine learning and transfer learning. The main idea behind domain adaptation is to learn from a fine-tuned model in source domain on different target domain. The two domains must have some similarity so that informative knowledge can be transferred from source domain to target domain. Several variants of domain adaptation approaches have been conducted on downstream NLP tasks such as semantic classification (Blitzer et al., 2007) and natural language parsing (McClosky, 2010). However, the work on domain adaptation at word embedding level is sparse. Lu et al. proposed a general regularization framework for domain adaptation in 2016 (Lu et al., 2016), where the parameters from multiple domains are constrained to be learnt jointly and remain similar to each other, in order to ensure transfer between multiple domains. Meanwhile, the model has hyper-parameters to determine the amount of information transfer between domains.

Given text corpus $D^1, D^2, ..., D^N$ from N domains, parameter vectors w_i in domain i and hyperparameter scalar λ , the objective function under this framework is:

$$L = \sum_{i=1}^{N} L_i(D^i; w_i) - \sum_{i=1}^{N} \lambda_{0,i} \|w_i\|^2 - \sum_{1 \le j < k \le N} \lambda_{j,k} \left\| w_j - w_k \right\|^2 \tag{2.1}$$

where $L_i(D^i; w_i)$ is the traditional machine learning objective function in domain i. $\lambda_{0,i} \|w_i\|^2$ is a L2-regularization term to prevent overfitting, and $\lambda_{0,i}$ is a positive scalar for weight decay. $\lambda_{j,k} \left\| w_j - w_k \right\|^2$ is another regularization term to keep parameter vectors in different domains close to each other. $\lambda_{j,k}$ is hyperparameter controlling the amount of transfer for each parameter pair.

Domain word embeddings can be improved by incorporating knowledge from multiple domains under this general regularization framework. (Bollegala et al., 2015) proposed an unsupervised algorithm for learning cross-domain word embeddings. They used common words appear in multiple domains as pivots, and build word vector model to predict surrounding 'non-pivots' using pivots, meanwhile making pivots having similar word embeddings in multiple domains by adding regularization terms. They demonstrated that word embeddings learned under this framework can capture batter domain semantic relations and make domain-insensitive word embeddings more accurate. (Yang et al., 2017) proposed similar algorithm with Bollegala, but they also learn word embeddings for words which are neither pivots nor non-pivots.

Chapter 3

Corpora

In this section we describe one source domain and two target domains involved in downstream evaluation tasks. For the target domains, we discuss what they consist of, how they were collected, and the relevance of the vocabulary between source domain and target domain.

3.1 Source domain: Google News dataset

The Google News dataset is what google pre-trained Word2Vec model trained on. Google collected a subset of data from Google News including about 100 billions tokens. The pre-trained model contains 300-dimensional word vectors for 300 millions words from multiple domains.

3.2 Target domain 1: Quora question pairs dataset

The Quora question pairs dataset is a recent-released dataset from the large question answering sites Quora [1]. This dataset consists over 400,000 rows of question pairs, and binary labels indicate whether the two questions have essentially same meaning. The ground truth is labelled by human experts so that they might be inherently subjective. Although the dataset may include some amount of noise, authors of the dataset claimed that these labels are reasonable and correct in general. All of the samples are genuine from Quora. Figure 3.1 shows the first 10 rows of the dataset. We first extracted the textual data from the dataset and applied the text data pre-processing using Python. There are totally 6975371 tokens in this corpus. We deleted those rare words that

[1] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

occur less than 5 times in training corpus and replaced them with the UNK (unknown) token. Thus the vocabulary list is composed of 30300 unique words (including UNK).

We divide the Quora dataset into three parts: training, validation and test sets with a 80%-10%-10% split. Hence the training set contains 327472 question pairs for building classification models, the validation set contains 36386 question pairs for model tuning and the test set contains 36386 question pairs as well for testing the generalization performance of model.

0	0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
2	2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
3	3	7	8	Why am I mentally very lonely? How can I solve it?	Find the remainder when [math]23^{24}[/math] is divided by 24,23?	0
4	4	9	10	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?	Which fish would survive in salt water?	0
5	5	11	12	Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say about me?	I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?	1
6	6	13	14	Should I buy tiago?	What keeps childern active and far from phone and video games?	0
7	7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1
8	8	17	18	When do you use ã· instead of ã?	When do you use "&" instead of "and"?	0
9	9	19	20	Motorola (company): Can I hack my Charter Motorolla DCX3400?	How do I hack Motorola DCX3400 for free internet?	0

Figure 3.1: The first 10 rows of Quora dataset

Table 3.1 lists some statistics for the Quora dataset. The Quora corpus has totally 30300 unique words in vocabulary, 24963 (83.39%) of them have embeddings in google pre-trianed model. Actually, the dataset consists of questions from various fields and mix them without specific labelling. So is it reasonable to regard it as a domain-specific dataset? We computed the ratio of overlapping words among the subsets of source domain vocabulary (Google News) and target domain vocabulary (Quora). The subset is selected based on the top 10K frequent words in each corpus separately. Specifically, we choose a subset of 10K most frequently used words in source domain and a subset of 10K most frequently used words in target domain separately, and calculate how many words are commonly appeared in both subsets. We found that around half of words in Quora vocabulary are not frequently used in Google News vocabulary. These words may express important domain-specific significance but are not used as much in Google News corpus. Based on the different usage of words in different dataset, we can still think the Quora dataset is in ‘Quora domain’ and Google News dataset is in ‘Google News domain’.

Vocabulary size	30300
Number of words existing in pre-trained model	24963 (83.39%)
Number of overlapping words in top 10K frequent word list	5075 (50.75%)

Table 3.1: Key statistics for the Quora dataset

3.3 Target domain 2: PubMed RCTs dataset

PubMed is a free search engine that access the references and abstracts related to life science and biomedical topics. The PubMed 200k RCT dataset is presented by (Dernoncourt and Lee, 2017), which consists of about 200k abstracts of randomized controlled trials (RCTs) in PubMed, and totally 2.3 millions sentences labeled with their role in the abstract. The labels include: background, objective, method, result and conclusion. All numbers in the dataset are replaced by '@' sign. The author also released a subset of PubMed 200k that is PubMed 20k. Figure 3.2 illustrates an example of one abstract composed by several labelled sentences, as formatted in the PubMed RCTs dataset.

We divide the Pubmed dataset into three parts: training, validation and text sets with a 80%-10%-10% split. Hence the training set contains 180037 short sentences for building classification models, the validation set contains 30212 short sentences for model tuning and the test set contains 30135 short sentences for testing the generalization performance of model.

Table 3.2 lists some statistics for the Pubmed dataset. The Pubmed corpus has totally 27188 unique words in vocabulary, and 16868 (62.04%) of them have embeddings in google pre-trianed model. This amount is smaller than Quora vocabulary. We also computed the ratio of overlapping words among the subsets of source domain vocabulary (Google News) and target domain vocabulary (Pubmed) using the same method we described in section 3.2. We observed that the PubMed corpus has only around 30% overlapping words with Google News corpus. It indicates that Pubmed dataset contains more domain-specific words that only be frequently used in Pubmed domain.

###24293578
OBJECTIVE To investigate the efficacy of @ weeks of daily low-dose ...
METHODS A total of @ patients with primary knee OA were randomized ...
METHODS Outcome measures included pain reduction and improvement ...
METHODS Pain was assessed using the visual analog pain scale (@-@ mm) .
METHODS Secondary outcome measures included the Western Ontario ...
METHODS Serum levels of interleukin @ (IL-@) , IL-@ , tumor necrosis ...
RESULTS There was a clinically relevant reduction in the intervention ...
RESULTS The mean difference between treatment arms (@ % CI) was ...
RESULTS Further , there was a clinically relevant reduction in the serum ...
RESULTS These differences remained significant at @ weeks .
RESULTS The Outcome Measures in Rheumatology Clinical ...
CONCLUSIONS Low-dose oral prednisolone had both a short-term and ...

Figure 3.2: Example of one abstract with labelled short sentence in Pubmed corpus

Vocabulary size	27188
Number of words existing in pre-trained model	16868 (62.04%)
Number of overlapping words in top 10K frequent word list	3314 (33.14%)

Table 3.2: Key statistics for the Pubmed dataset

Chapter 4

Word vector models

In this section, we first present the general data pre-processing procedure. Then we introduce all kinds of models we choose for word embeddings learning and tuning.

4.1 Text data pre-processing

Data pre-processing is an important procedure between collecting text corpus and building a robust NLP system. A text data pre-processing pipeline includes three major steps: Tokenization, noise removal and normalization. Tokenization is the process of split a string of input sentence into tokens. For example, in the text string: `What is the step by step guide to invest in share market ?` The raw input of sentence should be explicitly split into 13 tokens: `What, is, the, step, by, step, guide, to, invest, in, share, market,?` The resulting tokens are then passed on to next steps of processing. The steps after tokenization are thus working on a word level instead of a text level.

The noise removal task involves removing text file headers or footers, HTML and XML tags, JSON formats, etc. Normalization refers to a series of tasks include: converting all text to the same case (upper or lower case), removing punctuations, non-English characters or signs, stop words, etc. Stop words are most commonly used words in a language (i.e. 'the', 'a', 'so'). They usually connect parts of a sentence rather than showing significant information. Noise removal and normalization are much more task-specific than tokenization. The boundary between noise removal or normalization and data collection is a blurred one. We usually empirically determine the words we reserve given specific text corpus.

There we have a walkthrough of the text data pre-processing framework. We will

use the same process to extract and clean the textual data from raw dataset foremost before implementing any downstream NLP model.

4.2 Baseline 1: Google pre-trained Word2Vec model

Some research organizations have already published pre-trained word embedding models that trained on very large text corpora under the Public Domain Dedication and License. These word vector models can be downloaded and incorporated into our models to help us produce better domain word vectors. One example of this model is Googles Word2Vec model. They published a set of 300-dimensional pre-trained word vectors for 3 millions words and phrase, which trained on part of Google News dataset including roughly 100 billions tokens. Such considerable training corpora allow huge number of words converge to informative vectors. Since the pre-trained models are trained on different domains, they can introduce more text data that cover additional useful information, which can help consolidating the word embeddings learning in target domain.

In this paper, we utilized the pre-trained embedding set in three ways:

(1) Simply use the pre-trained word embeddings as features to a model for specific tasks;

(2) Consider the pre-trained embedding set as parameters from source domain. We train a different set of word embeddings on target domain under the regularization domain adaptation framework, and use the learnt word embeddings as features to a model for specific task.

(3) Initialize a word vector model with pre-trained embeddings, then retrain the word embeddings in target domain, and use the learnt word embeddings as features to a model for specific task.

(4) Concatenate the pre-trained embedding set with other embedding sets we trained on target domain corpus, and use the concatenated embeddings as features to a model for specific tasks;

4.3 Train word embeddings on domain corpus

In most of previous work, in order to yield high quality embedidngs, the influence of the corpus domain is considered to be dominant (Lai et al., 2016). Here we include

the basic Skip-Gram model for training word embeddings and some variants. All the models take as input training samples generated from target domain corpus.

4.3.1 Baseline 2: Skip-Gram with negative sampling

The model we used to train word embeddings on target domain corpus is Skip-Gram model with negative sampling. The Skip-Gram is one of Word2Vec models proposed by Mikolov (Mikolov et al., 2013b) that try to represent each word in a corpus with a fixed length vector, and making words that have similar syntactic or semantic meanings also be closed to each other in this vector space.

The main idea behind Skip-Gram is given a specific word in a sentence from a large corpus, the model try to predict the nearby context words of the specific word through a two layer neural network. The hidden layer is represented by a weight matrix with n rows and d columns, where n is the vocabulary size and d is the word vector dimension. Thus the rows of the this weight matrix are actually the word vectors (word embeddings) that we need. The output layer is a softmax regression classifier that calculate the probability for every word in vocabulary of being the context word. The goal of the model is to learn the hidden layer weight matrix, and the output layer will be removed after training.

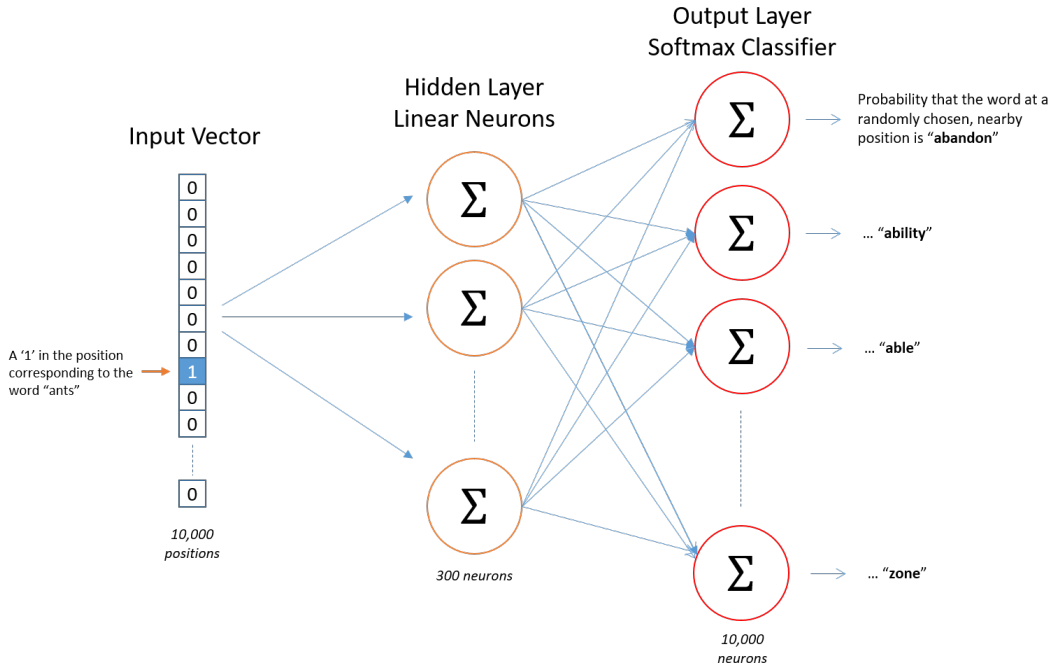

Figure 4.1: Skip-Gram model architecture

The Skip-Gram model architecture is shown at Figure 4.1. Assume that there are 10,000 unique words in our vocabulary, and each word is represented by a 300-dimensional vector. Hence the hidden layer is a $10,000 \times 300$ weight matrix shared for all words in the same way, and the output layer also has 10,000 neurons for every word in vocabulary. We first represent an input word w as a one-hot vector, and look up its corresponding row in weight matrix to . Then we multiply the word vector $\mathbf{w}$ for word w extracted from hidden layer against weight vector $\mathbf{c}_i$ in each output neuron separately. The softmax classifier applies the function $exp(\cdot)$ to the result in each output neuron, and divide the result by the sum of all 10,000 output results, in order to make the outputs to sum up to 1. Hence the final result $p(c_j|w)$ in equation (4.1) is the possibility of word c_j nearby input word w:

$$p(c_j|w) = \frac{exp(\mathbf{c}_j \cdot \mathbf{w})}{\sum_{i=1}^{N} exp(\mathbf{c}_i \cdot \mathbf{w})} \tag{4.1}$$

where $\mathbf{w}$ is word vector for input word w comes from weight matrix in hidden layer, $\mathbf{c}_j$ is weight vector in output layer for word c_j, N is vocabulary size.

We train the model by feeding word pairs into neural network. A word pair consists of an input word (current word) and a target word (context word) that we extracted from sentences in our training corpus. The number of context words we select for each specific word is determined by a hyperparameter called 'window size'. For example, here is a sentence: 'ants in my garden'. I use a small window size of 2. Thus the training samples are: (ants, in), (ants, my), (in, ants), (in, my), (in, garden), (my, ants), (my, in), (my, garden), (garden, in), (garden, my).

The training objective of Skip-Gram model is to minimize the negative log probability of actual target word w_O:

$$L_D = -logp(w_O|w_I) \tag{4.2}$$

4.3.1.1 Negative sampling

The original Skip-Gram model applies softmax classifier among every word in vocabulary, which means it updates all of the weights in the output layer for every training sample. Mikolov (Mikolov et al., 2013a) proposed a more efficient way of updating the weights using the negative sampling approach. With negative sampling, we randomly select a small subset of 'negative' words together with one 'positive' word to update the weights for. For example, when training on the word pair (ants, garden), the input

word is 'ants', the positive word is 'garden', and all of other words in vocabulary are negative words. The network makes the vector of positive word more similar to input word than the vector of other randomly chosen negative words. In the embedding layer, only the weights for input word are updated regardless of using negative sampling or not. The network architecture is same as original Skip-Gram model but the training objective of Skip-Gram with negative sampling is modified to minimize:

$$L_D = \sum_{(w,c)\in D} \#(w,c)\,(log\sigma(\mathbf{w}\cdot\mathbf{c})) + \sum_{i=1}^{k} E_{c_i^{'}\sim P(w)}\left[log\sigma(-\mathbf{w}\cdot\mathbf{c}_i^{'})\right] \tag{4.3}$$

Where D refer to the domain corpus from which we train the word embeddings. $\mathbf{w}$ is the word vector for current word w, $\mathbf{c}$ is the word vector for positive word (context word) c, $\mathbf{c}^{'}$ is the word vector for a negative word $c^{'}$ randomly selected from corpus using a unigram distribution $P(w)$. We sampled k negative words from dataset. $\#(w,c)$ is their number of co-occurrence in corpus D. The function $\sigma(\cdot)$ is the sigmoid function. The first term makes actual target word vector more similar to input word vector, and the second term makes negative word vector far from input word vector.

In this paper, we applied Skip-Gram model with negative sampling as our second baseline model for word embedding learning on target domain corpus.

4.3.2 A regularization-based skip-gram model

The basic Skip-gram model is constrained to the target domain where a specific task involves. One question is now that we had the pre-trained word embeddings learnt on different domains, can we utilize them to guide our embedding learning in target training corpus, so as to consolidate the word embeddings in target domain? The answer is yes. This problem can be solved by domain adaptation that transfer domain-independent knowledge between domains.

We modified the basic Skip-Gram model to a regularization-based Skip-Gram model, which is inspired by the domain adaptation framework proposed by (Lu et al., 2016). This approach encourages parameters in multiple domains being closed to each other, so that knowledge from source domain can be transferred to target domain. In our models, the source domain is text corpus which pre-trained word embeddings trained on, and the target domains are text corpus which our downstream tasks based on. One major question is how to determine the quantity and type of knowledge that is worth learning from source domain for target domain. To address the problem, Bollegala (Bollegala et al., 2015) proposed a method to select a subset of words that occur in

both domains to act like 'pivots'. We hypothesis that vectors of pivots that trained on the source domain represent domain-independent meanings, so that they might encode common semantic information on the target domain. Then we build model to ensure that vector of pivots have similar value in two domains. To be specific, we first select a set of pivots which are frequent words common to both domains. Then we add a regularization term for the pivots to the objective when training their embeddings, and train the non-pivots embeddings in the meantime. Therefore we learn the word embeddings on target domain corpus as follows:

$$L'_{D_t} = L_{D_t} + \sum_{w \in D_t \cap D_s} \alpha_w \cdot \|\mathbf{w}_t - \mathbf{w}_s\|^2 \tag{4.4}$$

where L_{D_t} refer to the objective function of basic Skip-Gram model trained on target domain, $\mathbf{w}_t$ is the word embeddings for pivot w in target domain, $\mathbf{w}_s$ is word embeddings for pivot w in source domain from pre-trained model, α_w are hyperparameters which refer to the amount of transfer from source domain to target domain for each pivot. Note that in the regularized objective, only the pivots are considered in regularization term. The select range of pivots are all of words that appear in both source domain and target domain. We sorted the pivots in decreasing order by frequency in target domain and tuned the number of pivots in Appendix A.

We used two ways to define the hyperparameter α_w, one is assigning a constant to α_w, which means every pivot is constrained with same amount of transfer across domains, regardless of individual difference. Another is defining a significance function $\phi(w)$ that control the amount of transfer for each pivot. The amount of α_w are defined as follows:

$$\alpha_w = \lambda \cdot (\sigma(\phi(w)) - 0.5) \tag{4.5}$$

where λ is a hyperparameter to control the magnitude. The significance function $\phi(\cdot)$ controls the degree of knowledge transfer from source domain to target domain. In order to define the $\phi(\cdot)$, we first define the frequency of pivot w in the corpus D as $f_D(w)$, which refer to the number of times the pivot w occurs in the domain corpus D. Thus we can define the normalized frequency for the pivot w as follows:

$$F_D(w) = \frac{f_D(w)}{\sum_{w \in D_t \cap D_s} f_D(w')} \tag{4.6}$$

Based on this we can define the function $\phi(\cdot)$, which is inspired by the well-know Srensen-Dice coefficient (Sørensen, 1948) (Dice, 1945), as follows:

$$\phi(w) = \frac{2 \cdot F_{D_s}(w) \cdot F_{D_t}(w)}{F_{D_s}(w) + F_{D_t}(w)} \tag{4.7}$$

Note that the value of $\phi(w)$ is affected by both $F_{D_s}(w)$ and $F_{D_t}(w)$, so the $\phi(w)$ will be high only if the pivot w is a frequent word in both two domains. Then we apply sigmoid function $\sigma(\cdot)$ to compresses the $\phi(w)$. Since that the value of $\phi(w)$ is positive, the result of $\sigma(\phi(w))$ is always greater than 0.5, so we minus 0.5 after applying sigmoid function.

4.3.3 Retrain pre-trained embeddings

In this case, we still train the basic Skip-Gram model on target domain corpus, but initialize the embedding layer with pre-trained embeddings instead of random initialization. This is also a kind of transfer learning that retrain the weights in a new space, to make use of the full knowledge form source domain.

4.4 Combination of different embedding sets

So far, each of the individual models we explained in section 4.2 will be built and fine-tuned for optimal hyperparameter setup in specific tasks in following sections. Here we consider a simple ensemble method to test whether combining two of them can capture the strengths of each. Ensemble method is a machine learning technique that combine several models for better predictive performance than any single model. The use of ensemble methods are found to make significant improvement in many fields of machine learning. The combination way we test is concatenating word embedding for a given word across pre-trained model and other model trained on target domain. Hence the concatenated word embbeddings will be:

$$w_{CAT} = [w_1, w_2] \tag{4.8}$$

4.5 Dimensional reduction

Google's pre-trained Word2Vec model includes 300-dimensional word embeddings. Dealing with such high dimensional features is problematic because it can make high computational cost and large memory requirements in practical use. Hence we use

the PCA technique to decompose the pre-trained vectors from 300-dimensional to 50-dimensional, while preserving most of the useful information in data. All embeddings we intend to train on domain corpus are 50-dimensional as well. On the other hand, we also use the PCA technique to decompose the concatenated vectors in section 4.4 from 100-dimensional to 50-dimensional, so as to to ensure its dimensionality is consistent with other word embedding sets we evaluated.

PCA is abbreviation for Principal Component Analysis (Shlens, 2014), which is a useful statistical procedure to reduce the dimension of a data by only keeping the most important dimensions and removing other dimensions. The detailed working of PCA is:

1) Compute the covariance matrix of data points (vectors) in continuous space.

2) Compute the eigen vectors and corresponding eigen values of the covariance matrix.

3) Sort the eigen vectors according to their eigen values in decreasing order.

4) Select the top K eigen values, K is the size of dimension we decompose to.

5) Transform the original vectors into k-dimensional by keeping only the dimensions corresponding to the top K eigen values.

Performing dimensional reduction technique may potentially lose some information of data. We use the standard word similarity benchmarks summarized by (Faruqui and Dyer, 2014) for investigating the impact of decomposing the concatenated word embedding from 100D to 50D using PCA. The datasets in word similarity benchmarks contains word pairs that have been given similarity rankings by humans. While evaluating the word embedding sets, we computed the cosine similarity of their word vectors. Then, we calculated the Spearman's rank correlation coefficient between the human ranking and ranks computed by certain word embedding set. The correlation value will be higher if word embedding set captures better word semantic relationship.

The embedding sets we used for evaluation include an original 100D embedding set, who concatenates a set of 50D pre-trained word embeddings provided by google with a set of 50D word embeddings trained in Quora domain using regularization-based Skip-Gram model (in Section 4.3.2), and its 50D version decomposed by PCA.

Table 4.1 summarizes the results across two embedding sets on 8 datasets. The 50D embeddings have an average performance decline of 0.0181 across 8 datasets. This result shows that the trade-off between preserving information and reducing dimensionality is acceptable.

Dataset Name	Pairs found	Correlation (100D)	Correlation (50D)	Decline
WS-353	329	0.6149	0.5952	0.0197
WS-353-SIM	186	0.7015	0.6891	0.0124
WS-353-REL	240	0.5857	0.5644	0.0213
MEN	2635	0.6694	0.6339	0.0355
MTurk-287	252	0.6721	0.6639	0.0082
MTurk-771	755	0.6053	0.5838	0.0215
SimLex-999	952	0.2527	0.2339	0.0188
SimVerb-3500	3084	0.1723	0.1647	0.0076

Table 4.1: Spearman's correlation coefficient across 100D and 50D embedding sets

Chapter 5

Evaluation

Many NLP tasks use word embeddings as input features to improve their performance. Here we use two downstream classification tasks, in which the word embedding is the only feature, to evaluate the quality of individual word embedding sets produced by different word vector models. The word vector models we include for evaluation are:

PRE The pre-trained Word2Vec model trained on Google News domain, released by Google. The original word vectors are 300-dimensional and we reduce the dimensionality to 50-dimensional for the sake of less computational cost.

RET Retrain the Skip-Gram model using 50-dimensional pre-trained word embeddings for weight initialization in embedding layer.

BAS The basic Skip-Gram model using 50-dimensional random initialization in embedding layer.

REG The regularization-base Skip-Gram model using 50-dimensional random initialization in embedding layer.

CAT The concatenation of the PRE and REG model. The original concatenation is 100-dimensional and we reduce it to 50-dimensional.

5.1 Task 1: Text-Pair Classification

Our first experiment was conducted on text pair classification. Text pair classification is an important type of NLP problem, especially the natural language inference problem. This type of question lets a model assigns labels to text-pairs, based on some relationship between them. When the relationship is symmetric, it can be used to detect whether two questions are duplicate or not. Previous works have been conducted on the Stanford Natural Language Inference (SNLI) corpus released by (Bowman et al.,

2015), which provide over 500,000 pairs of short sentences. However, the SNLI dataset is too much artificial, hence the data is less possible to occur in real applications. Fortunately, a recent-released dataset from the large question answering sites Quora is accessed online by anyone. This dataset consists of over 400,000 lines of question pairs, and binary labels indicate that whether the two questions have essentially the same meaning. Our task is to predict the binary label for two input questions.

5.1.1 Neural network architecture

The neural network model architecture we used in this task is based on the SNLI benchmark (Bowman et al., 2015), which is a standard approach that people use to address semantic inference problem. We adapted the original neural network architecture to tackle the problem in Quora domain. Figure 5.1 depict the neural network architecture, which is:

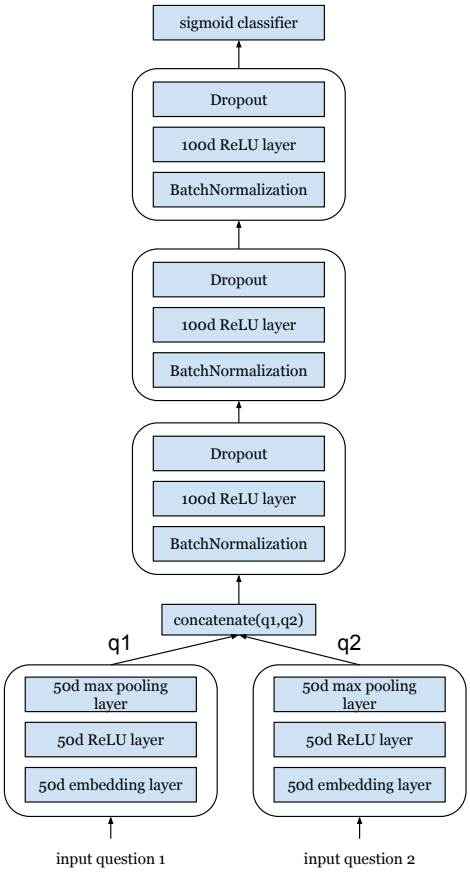

Figure 5.1: The neural network architecture for question-pair classification task

- take two questions as inputs separately

- extract 50D word vectors from embedding layer for each sentence
- pass 50D word vectors through a ReLU translation layer
- use max operator to select the maximal value among all words in each dimension, which convert a sequence of word vectors into a single sentence vector
- concatenate the two 50D sentence vectors into a 100D vector
- pass the 100D vector through a stack of four 100D fully-connected ReLU layer
- pass the 100D vector through output layer with sigmiod binary classifier

The embedding layer is initialized with different word embedding sets we want to evaluate, which are learned using methods we mentioned in section 4, and will not be trained during training excepted retrained model. All of other parameters are randomly initialized and trained using binary cross-entropy loss and Adam minibatch SGD for optimizer. We run training for 50 epoches and choose the best performance on validation set. Dropout is applied additionally to the output layer and fully-connected layer with a fixed dropout rate 0.1 to reduce overfitting.

5.1.2 Word2Vec model setup

hyperparameter	choice
Corpus domain	Quora
Corpus size (M-token)	[0.01, 0.05, 0.1, 0.5, 1, 3, 5]
Epoch number	50
Batch size	1000
Window size	5
Embedding dimensionality	50
Negative samples number	100
learning rate	0.01
Optimzer	SGD + Adam
Vocabulary size	30300

Table 5.1: basic Skip-Gram model setup in task 1

hyperparameter	choice
Corpus domain	Quora
Corpus size (M-token)	[0.01, 0.05, 0.1, 0.5, 1, 3, 5]
Epoch number	50
Batch size	1000
Window size	5
Embedding dimensionality	50
Negative samples number	100
learning rate	0.0001
Optimzer	SGD + Adam
Vocabulary/pivots size	30300/24965
α_w	$10 \cdot \sigma(\phi(w) - 0.5)$

Table 5.2: regularization-based Skip-Gram model setup in task 1

Table 5.1 lists the fine-tuned hyperparameter setup we used for basic Skip-Gram model and retrained Skip-Gram model. We did not tune the hyperparameters in these models. Meanwhile, we kept the value of some common hyperparameters fixed in latter models.

Table 5.2 lists the optimal hyperparameter setup for regularization-based Skip-Gram model. We tuned the choice of α_w with grid search. All of the results of hyperparameter tuning are put in Appendix A. We found that using all overlapping words as pivots and applying significance function with $\lambda = 10$ to define the amount of transfer yield best word embddings in this case.

5.1.3 Results and analysis

We first investigate how the target corpus size effect the quality of word embeddings. From the result in Figure 5.2 and Table 5.3 we can conclude that the larger target corpus is always superior to the smaller one. When the corpus size is lager than 1M-token, training basic Skip-Gram models (BAS) or retrain Skip-Gram models (RET) in target domain can yield better word embeddings than just using pre-trained (PRE) one. The threshold of target corpus size decrease to 500K-token if we use the regularization-based Skip-Gram model (REG). Concatenated models are consistently superior to the

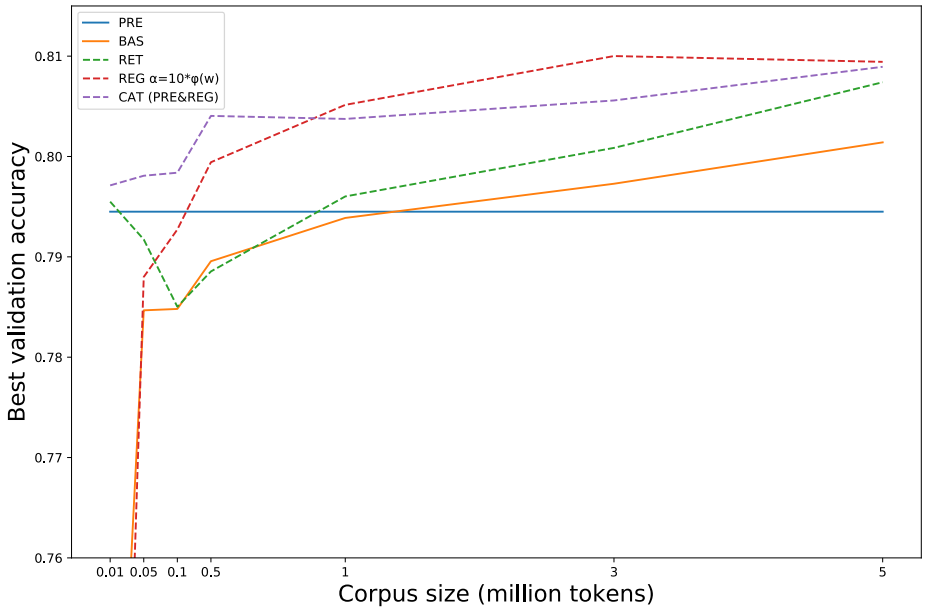

Figure 5.2: Best validation accuracy for embedding sets using different word vector models trained on subsets of the Quora corpus from 10K-token to 5M-token

pre-trained model even if the corpus size is small. Moreover, the three non-baseline models generally outperform the basic Skip-Gram model. We observed improvement of approximately 1% using regularization-based model and concatenated model. The performance boost of retrain model is lesser than other two models.

Summarizing the above results we can draw conclusions that 1M-token in this case seemed to be a boundary to determine whether the target domain corpus is large enough to produce good word embeddings. When training corpus is smaller than 1M-token, using a concatenation of regularization-based model with pre-trained model can yield better word embeddings. While using regularization-based model individually is good enough for embedding learning when target domain corpus is larger than 1M-token.

Then we concentrated on some specific samples to have a better understand on how in-domain training and knowledge transfer influence the shape of word embeddings. We investigate word embedding sets that trained on 5M-token corpus using the pre-trained model (PRE), basic Skip-Gram model (BAS) and regularization-based model (REG) respectively.

In the Quora question dataset, many question pairs are the same for most of the content while only a few words are different, which exactly indicate whether the ques-

Model	5M	3M	1M	500K	100K	50K	10K
PRE				79.45			
RET	80.74	80.09	79.60	78.85	78.50	79.17	79.55
BAS	80.14	79.73	79.39	78.96	78.48	78.47	71.86
REG	**80.94**	**81.00**	**80.51**	79.94	79.27	78.80	68.37
CAT	**80.89**	80.56	**80.37**	**80.49**	**79.84**	**79.81**	**79.71**

Table 5.3: Best validation accuracy for embedding sets using different word vector models trained on subsets of the Quora corpus from 10K-token to 5M-token

tion pair is duplicate or not. Therefore, we tried to find out the factors that influence the model prediction by comparing the cosine similarity score between these word pairs computed in different word embedding sets.

Cosine similarity score measures the similarity of two vectors $\vec{a}$ and $\vec{b}$ by calculating the cosine of the angle between them. It can generate a value that indicate how related are two words by look at their angle rather than magnitude. The cosine similarity formula is:

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \left\|\vec{b}\right\|} \tag{5.1}$$

From the samples in Table 5.4 we can find that the pre-trained model is insensitive to the difference of numbers. This is because for most NLP tasks, numbers are open word classes and not very meaningful, so that they usually be replaced with meta tag to reduce vocabulary size. However, in the Quora corpus, numbers usually represent years, quantities, mathematical numbers or currencies, which can be essential to distinguish two different questions. Word embeddings learned from in-domain corpus address this problem by training several meaningful numbers individually, such that they could be identified in downstream tasks.

Question 1	Question 2	Lebel	PRE	BAS	REG
Should I buy a **2015** Macbook Pro or wait for the new updated version?	Should I buy a MacBook Pro on March 18th, **2010** or wait for updates?	0	1	0	0
`cosin_similarity`(2015, 2010) =			1	0.551	0.519
What are three numbers that are multiplied to get -**125**?	What two numbers multiply to get -**240**?	0	1	0	0
`cosin_similarity`(125, 240) =			1	0.613	0.314
What are the best in ear headphones under **1500**?	What are good in ear headphones under **250**?	0	1	0	0
`cosin_similarity`(1500, 250) =			1	0.582	0.565

Table 5.4: Samples for PRE-incorrect BAS-correct REG-correct trained on 5M-token corpus

On the other hand, the pre-trained model can introduce more text data containing useful contextual information from source domain. From the samples in Table 5.5 we can find that the similarity of words changes from basic Skip-Gram model to regularization-based Skip-Gram model. Moreover, the intervention of pre-trained model makes some of similarity value in regularization-based model become intermediate value between pre-trained model and basic Skip-Gram model. Some words, for example (fever, constipation) in Table 5.5, represent different meanings in questions. Table 5.6 lists their nearest neighbors in different embedding sets computed by similarity. We can infer that 'fever' and 'constipation' would be in different contexts in the Google News Corpus because they have totally different similar words. Meanwhile, they share two common nearest words 'pain' and 'throat' in basic Skip-Gram model. Given that 5M-token corpus is large enough for word embedding convergence, the unreasonable similarity between 'fever' and 'constipation' may be due to the special format of Quora dataset that contains plenty of repeated sentences with just one or two tokens replaced by different words. Figure 5.3 gives some examples showing that many different words frequently appear in similar contexts. As we mentioned in

Section 2.3.2, Word2Vec models follows the distributional hypothesis that make words having similar contexts tend to have similar vectors. The basic Skip-Gram model, as one of the Word2Vec model, might be misled to learn similar word representations for these words. As a consequence, the downstream model mistakenly believes that these questions have same meaning. In addition, some words, for example (you, us) and (antivitus, malware) in Table 5.4, represent similar meaning in questions. However, because of the text limitation, some similar words do not have enough common contexts, which prevents them from converging to be nearby in the vector space, hence predict incorrect labels in specific downstream task. Finally, regularization-based model addresses these problems by incorporating more contextual information from pre-trained model, such that words having different meanings are far apart in the vector space, while similar words are close to each other. We concluded that using pre-trained model to guide word embedding learning in target domain can capture more semantic relationship when training corpus has limited usage of words.

...
What's the remedies for constipation?
What are some home remedies for constipation?
What are some herbal remedies for constipation?
What are some home remedies for fever?
What are some home remedies for treating diarrhea?
What home remedies help with nausea?
What are the home remedies for cold?
...

Figure 5.3: Examples of some similar questions in Quora corpus

Question 1	Question 2	Lebel	PRE	BAS	REG
What is the future of **Ruby** and **Python**?	What is the future of **Python**?	0	0	1	0
`cosin_similarity`(ruby, python) =			0.344	0.775	0.588
What were some of the rarely mentioned factors leading to the rise of **Obama**?	What were some of the rarely mentioned factors leading to the rise of **Putin**?	0	0	1	0
`cosin_similarity`(obama, putin) =			0.647	0.766	0.724
What are some home remedies for **fever**?	What are some home remedies for **constipation**?	0	0	1	0
`cosin_similarity`(fever, constipation) =			0.486	0.687	0.611
What are the **best** kitchen knives for a beginner??	What are the **good** kitchen knives?	1	1	0	1
`cosin_similarity`(best, good) =			0.879	0.610	0.839
Why is sugar bad for **you**?	Why is sugar bad for **us**?	1	1	0	1
`cosin_similarity`(you, us) =			0.667	0.109	0.485
Why we need **Antivirus** for computer?	Why do I need **malware** protection?	1	1	0	1
`cosin_similarity`(antivirus, malware) =			0.865	0.416	0.707

Table 5.5: Samples for PRE-correct BAS-incorrect REG-correct trained on 5M-token corpus

	PRE	BAS	REG
fever	spasm, fright, mania, itch, sensation, hangover, chills, anxiety	**pain**, sleeping, stomach, headache, teeth, shower, sick, **throat**	cold, fainting, cough, headaches, stomach, sick, dry, dizzy
constipation	hemorrhoids, nausea, reflux, diarrhea, insomnia, rhinitis, ulcers, bloating	diarrhea, miscarriage, fatigue, infection, **pain**, allergies, cramps, **throat**	diarrhea, cramps, insomnia, migraine, ADHD, diabetes, cough, gum

Table 5.6: Nearest neighbors of ‘fever’ and ‘constipation’ computed in different embedding sets

5.2 Task 2: Semantic heading Classification

The second evaluation task we carried on was semantic heading classification, which essentially a sentence classification task, where the goal is to label a sentence with its semantic role in the abstracts of randomized controlled trials (RCTs). The semantic roles include background, objective, method, result and conclusion. A RCT is a type of medical experiment in clinical research. When researchers search for previous clinical case, they usually have a quick look at the abstract to check whether the papers match their interest. The purpose of learning these labels is to extract structured information from original papers, in order to help researchers find out useful papers easily and quickly. There is no doubt that the specific task is in the field of medical science, and the usage of words in our training text corpus is limited to the medical context.

5.2.1 Network architecture

Figure 5.4 depict the neural network architecture, which is:

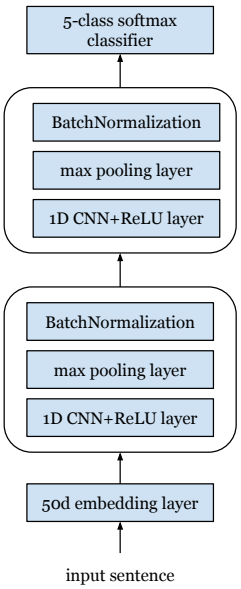

Figure 5.4: The neural network architecture for semantic heading classification task

- take a short sentence as inputs
- extract 50D word vector from embedding layer
- pass 50D word vectors through a stack of two CNN+ReLU+maxpooling layer

- a softmax layer for multi-class classification

The embedding layer is initialized with different word embedding sets and will not be trained during training. All of other parameters are randomly initialized and trained using softmax cross-entropy loss and Adam minibatch SGD for optimizer. We run training for 8 epoches and choose the best performance on validation set. L2 regularization is applied additionally on the CNN layers with weight decay = 0.001.

5.2.2 Word2Vec model setup

hyperparameter	choice
Corpus domain	Pubmed RCTs
Corpus size (M-token)	[0.01, 0.05, 0.1, 0.5, 1, 2, 3]
Epoch number	50
Batch size	1000
Window size	5
Embedding dimensionality	50
Negative samples number	100
learning rate	0.0001
Optimzer	SGD + Adam
Vocabulary size	27188

Table 5.7: basic Skip-Gram model setup in task 2

Table 5.7 lists the fine-tuned hyperparameter setup we used for basic Skip-Gram model and retrained Skip-Gram model.

Table 5.8 lists the optimal hyperparameter setup for regularization-based Skip-Gram model. We tuned the choice of α_w with grid search. All of the results of hyperparameter tuning are put in Appendix A. We found that using all overlapping words as pivots and assign α_w with fixed value 0.0001 yield best word embddings in this case.

hyperparameter	choice
Corpus domain	Pubmed RCTs
Corpus size (M-token)	[0.01, 0.05, 0.1, 0.5, 1, 2, 3]
Epoch number	50
Batch size	1000
Window size	5
Embedding dimensionality	50
Negative samples number	100
learning rate	0.0001
Optimzer	SGD + Adam
Vocabulary/pivots size	27188/16868
α_w	0.0001

Table 5.8: regularization-based Skip-Gram model setup in task 2

5.2.3 Results and analysis

From the results in Figure 5.5 and Table 5.9 we can find that the quality of word embeddings increase with the augment of training corpus. Basic Skip-Gram model (BAS) outperforms pre-trained model (PRE) while corpus size is larger than 1M-token. For regularization-based model (REG) and retrained model (RET), the boundary that gives better performance than pre-trained model is 500K-token. Concatenated model consistently outperforms per-trained model. These results are nearly same as previous conclusion we drawn in Quora domain. In general, peak performance for corpus size smaller than 500K-token is reached by concatenated model. While the regularization-based model gives best performance in larger corpus. All of the non-baseline models achieve better results than basic Skip-Gram model.

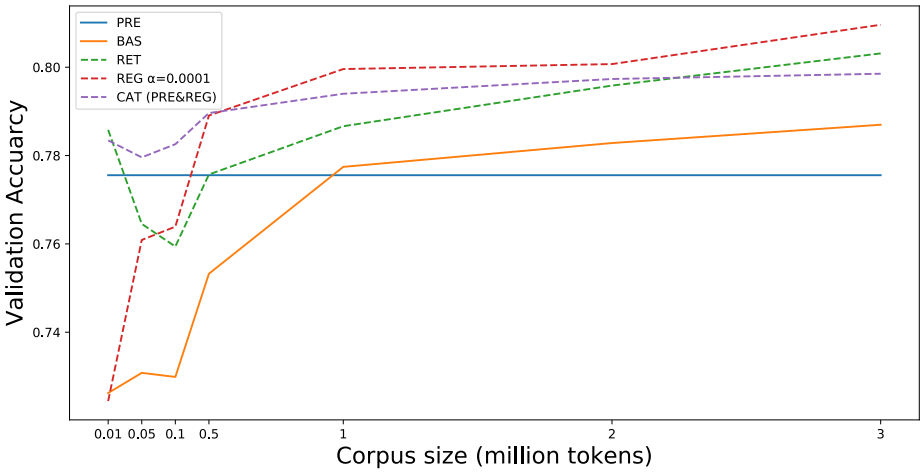

Figure 5.5: Best validation accuracy for embedding sets using different word vector models trained on subsets of the Pubmed corpus from 10K-token to 3M-token

Model	3M	2M	1M	500K	100K	50K	10K
PRE				77.56			
RET	80.31	79.58	78.66	77.57	75.94	76.45	**78.58**
BAS	78.70	78.28	77.74	75.32	72.99	73.08	72.62
REG	**80.96**	**80.07**	**79.96**	**78.91**	76.39	76.09	72.44
CAT	79.85	79.73	79.40	**78.96**	**78.26**	**77.96**	78.34

Table 5.9: Best validation accuracy for embedding sets using different word vector models trained on subsets of the Pubmed corpus from 10K-token to 3M-token

0: Background 1: Objective 2: Method 3: Result 4: Conclusion

Figure 5.10: Confusion tables for multi-class classification tasks, using four different word embedding sets trained on 1M-token Pubmed corpus as input features, best results on validation set.

Model	Background	Objective	Method	Result	Conclusion
PRE	0.538	0.582	0.795	0.767	0.602
BAS	0.509	**0.611**	0.785	0.783	0.619
RET	0.545	0.591	0.806	0.792	0.621
REG	0.552	0.598	**0.814**	**0.797**	0.639
CAT	**0.565**	0.591	0.812	0.795	**0.647**

Table 5.10: The kappa statistic for each category in different classifiers using 1M-token corpus for embedding learning, best trained model on validation set

Cohen's Kappa statistic is another measure for inter-rater agreement between two human raters. In machine learning, it is kind of classification accuracy but more useful in imbalanced data and multi-class problems, because it is normalized by expected accuracy. Figure 5.10 shows the confusion tables and kappa statistics of agreement between labels and predictions in validation set for different models. The corpus we used here is 1M-token. The columns correspond to one 'rater' reflecting the actual labels of each sentence to be classified (the ground truth), and rows correspond to another 'rater' reflecting the predictions of machine learning classifier. We found that the classes are not represented equally. The dataset consists of disproportionately high number of sentences for 'Method' and 'Result' classes. Nevertheless, some of the kappa scores are still consistent with the accuracy results reported in Figure 5.5 and Table 5.9. From the confusion tables we can also find that 'Objective' sentences are often confused with 'Background' sentences and 'Method' sentences are often confused with 'Result' sentences in PRE model, while they are less confused in BAS, REG and CAT models. On the contrary, 'Background' sentences are more confused with 'Objective' sentences in BAS model than in other models.

The overall kappa scores indicate that the three non-baseline models still outperform the baseline pre-trained model and basic Skip-Gram model. However, the difference between regularization-based model and concatenated model is smaller. To analysis agreement amongst ground truth and our classifiers in individual category, we compute the kappa statistic for the five semantic labels respectively. Results in Table 5.11 shows that regularization-based model gives better performance when assigning 'Method' and 'Result' labels, while concatenated model does better in 'Background' and 'Conclusion' class. Due to the imbalanced class distribution, classifiers is biased

to 'Method' and 'Result' classes when calculating the accuracy. This is why the overall accuracy score is higher than kappa score, and gap between regularization-based model and concatenated model is larger in accuracy score than in kappa score. Actually, the concatenated model and regularization-based model have almost the same improvement in general but different in particular aspects.

5.2.4 Limitation analysis

Although our three vector models indeed outperform baseline models, the improvement is not very pronounced. In this section we discuss several limitations in this task which prevent downstream model from obtaining better performance:

(1) The current neural network architecture is under a general text or sentence classification framework that classify input sentence in isolation without incorporating the preceding sentences. However, the semantic headings in an abstract from medical paper usually appear in sequence. For example, 'Background' usually appears in the first sentence and 'Conclusion' in last sentence. Hence using information from previous inputs may improve classification performance, especially for distinguishing 'Background' and 'Conclusion'. Since the goal of our work is to compare the quality of word embeddings generated by different word vector models, we did not put much attention on the optimization of network architecture.

(2) Some sentences are confusing that just not possible to achieve 100% accuracy even by a human expert. For example, 'Conclusion' usually summarizes how the 'Result' verifies or contradict the author's hypothesis. The boundary between 'Conclusion' and 'Result' is sometimes blurred. The sentence: *'Our meta-analysis, which included our own results and those of one other study , substantiated the results of the trial.'* is labelled with 'Result' by humans in dataset, but it is also reasonable to be labelled as 'Conclusion'.

Chapter 6

Conclusion

In this project, we fairly compared a series of approaches of learning high-quality word embeddings in specific domain. Although we did not find any particular models and settings that can yield best performance on all downstream tasks, we provided several guidelines for obtaining good word embeddings for domain-specific tasks.

A domain-specific corpus is a collection of texts where words have specific usage in that domain. We found that it is worthwhile to train a new word vector model on domain-specific corpus rather than using pre-trained word embeddings when downstream task is related to a certain domain, because many of words do not have word embeddings in pre-trained model. Another important factor of effecting domain word embeddings learning is the size of domain-specific corpus. Generally, large in-domain text data is essential to learn good domain word embeddings without information transfer from out-domain knowledge. Through our evaluations on two downstream domain tasks, we conclude that the boundary of corpus size which determines whether an in-domain corpus can independently yield better word embeddings than pre-trained model is 1M-token. Although it can not be arbitrarily thought as gold standard in any case, we can think it as an empirical standard for other downstream domain tasks.

Then we tested our three approaches that using pre-trained word embeddings to modify word vector modeling in domain specific corpus. They are retrained Skip-Gram model, regularization-base Skip-Gram model and concatenated model. They transfer the knowledge gained from pre-trained model to specific domain in different ways. Results on downstream tasks demonstrated that all of three models outperform basic Skip-Gram model. Meanwhile, concatenated model consistently outperforms pre-trained model. Retrained model and regularization-base model outperform pre-trained model only if larger in-domain text data is available, and the boundaries differ

in two tasks but are no larger than 1M-token as well. Overall, experiments shows that retrained model is the least competitive model among the three models. Concatenated model effectively improve word embeddings when in-domain corpus is limited that fail to individually produce good word embeddings. This also demonstrates that inferior word representations learned from limited data may still encode valuable in-domain information. However, concatenated model is less competitive than regularization-based model when we have access to more in-domain text data. The boundaries of corpus size that regularization method outperforms concatenation method are 1M-token and 500K-token in two evaluation tasks respectively. We hypothesize that 1M-token is an important threshold that determine which kind of word vector model is able to produce better word embeddings.

In an attempt to gain insight into how pre-trained model influences word embedding shaping in domain corpus, we tracked the similarity between certain words computed in different word embedding sets. We found that pre-trained word embeddings can help introduce more out-domain contextual information and incorporate them into domain word embedding learning when domain-specific corpus has limited usage of some general words.

Nevertheless, we do not suggest that concatenated model or regularization-based model is the only way to produce better domain word embeddings. In the future we intend to gain more insight into what kind of knowledge has been transferred from pre-trained embeddings to target domain. It is also important to investigate whether these approaches will work on other types of NLP tasks such as machine translation and question answering system.

Appendix A

Quora domain Word2Vec model tuning

Table A.1 lists the hyperparameter grid we tuned in the regularization-based Skip-Gram model. The training data is a subset of Quora corpus with 1M tokens.

hyperparameter	choice
Pivots number	[100, 300, 3000, 7000, all]
α (constant)	[0.1, 0.01, 0.001, 0.0001]
λ (in significant function)	[1, 10, 100]

Table A.1: hyperparameter grid in regularization-based Skip-Gram model

We tuned the amount of transfer in two ways:

(1) Define α as a constant, and tuned the number of pivots and value of α to obtain different word embedding sets. Then we use these word embeddings as features in Quora question-pair classification task. Figure A.2 and Figure A.3 show their performance in validation set. The optimal setup is considering all words that appear in both domains as pivots. The optimal choice of α is 0.0001.

(2) We define α using a significance function $\sigma(\cdot)$ and a hyperparameter λ. We tuned the value of λ among range of [1, 10, 20]. Figure A.4 shows their performance in Quora question-pair classification task. The optimal choice of λ is 10.

Figure A.4 compares the performance of best word embeddings in (1) and (2). We argue that using a significance function to define the amount of transfer for pivots between domains yield better word embeddings than assign same values to them in this case.

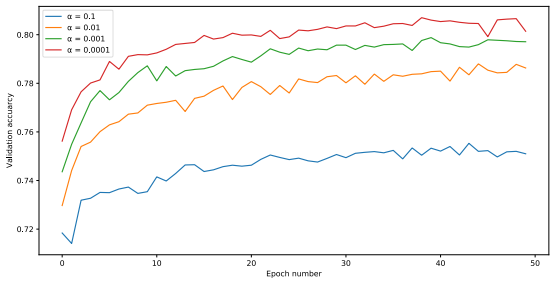

Figure A.1: Tuning constant α, 100 pivots

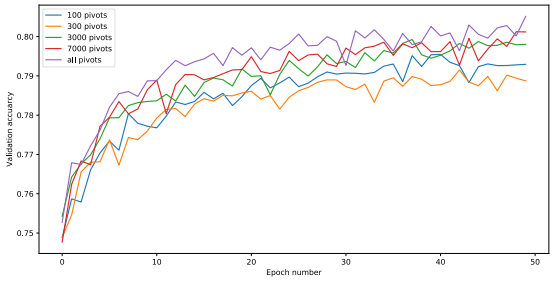

Figure A.2: Tuning pivots size, α = 0.0001

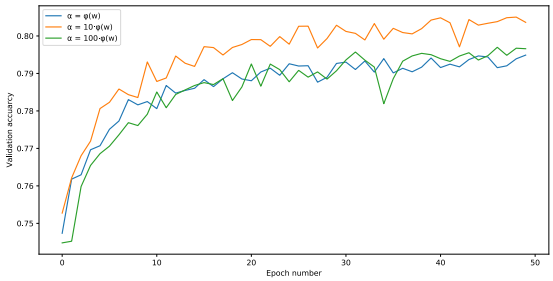

Figure A.3: Tuning λ in significance function

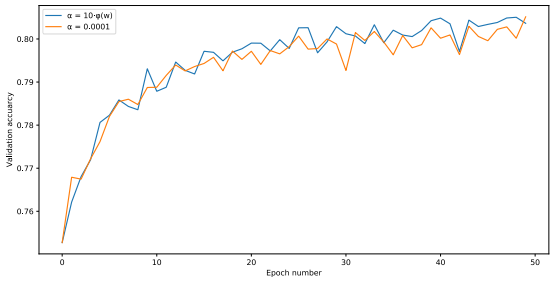

Figure A.4: Comparing two definitions of α

Appendix B

Pubmed domain Word2Vec model tuning

Table B.1 lists the hyperparameter grid we tuned in the regularization-based Skip-Gram model. The training data is a subset of Pubmed corpus with 1M tokens.

hyperparameter	choice
Pivots number	[100, 500, 1000, all]
λ (in significant function)	[1e3, 1e4, 1e5, 1e6]

Table B.1: hyperparameter grid in regularization-based Skip-Gram model

We tuned the amount of transfer in two ways:

(1) Define $\alpha = 0.0001$, which has been tuned in Appendix A, and tune the number of pivots to obtain different word embedding sets. Then we use these word embeddings as features in Pubmed sentence classification task. Figure B.1 shows their performance in validation set. The optimal setup is considering all words that appear in both domains as pivots.

(2) We define α using a significance function $\sigma(\cdot)$ and a hyperparameter λ. We tuned the value of λ among range of [1e3, 1e4, 1e5, 1e6]. Figure B.2 shows their performance in Pubmed sentence classification taskk. The optimal choice of λ is 1e5.

Figure B.2 also compares the performance of best word embeddings in (1) and (2). We argue that using a constant 0.0001 to define the amount of transfer for all pivots yield best word embeddings.

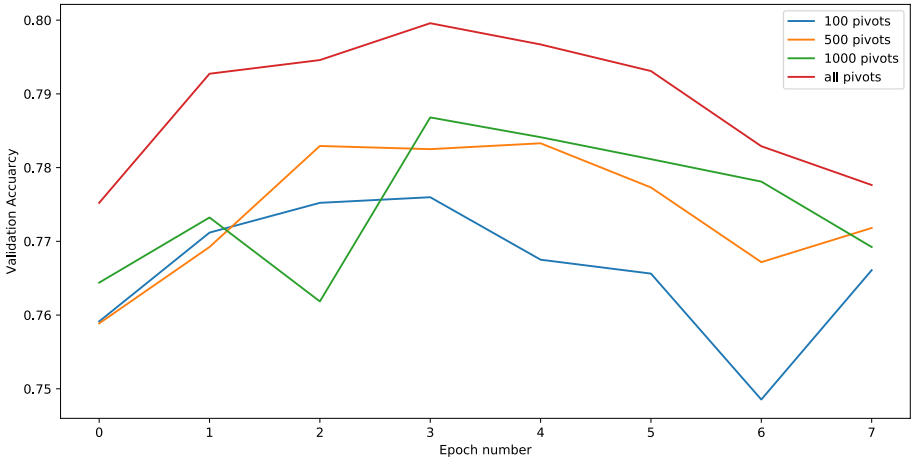

Figure B.1: Tuning pivots size, $\alpha = 0.0001$

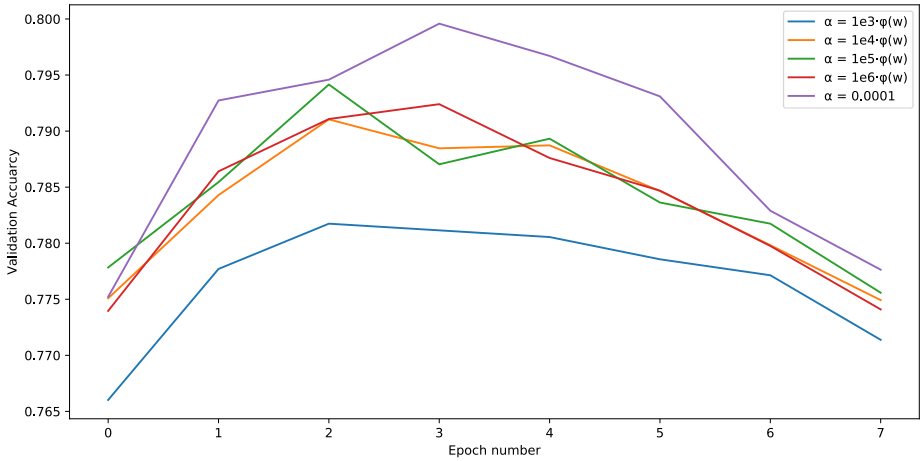

Figure B.2: Tuning λ in significance function

Bibliography

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

Bollegala, D., Maehara, T., and Kawarabayashi, K.-i. (2015). Unsupervised cross-domain word representation learning. *arXiv preprint arXiv:1505.07184*.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Dernoncourt, F. and Lee, J. Y. (2017). Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071*.

Dhillon, P., Rodu, J., Foster, D., and Ungar, L. (2012). Two step cca: A new spectral method for estimating vector models of words. *arXiv preprint arXiv:1206.6403*.

Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

Faruqui, M. and Dyer, C. (2014). Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA. Association for Computational Linguistics.

Garten, J., Sagae, K., Ustun, V., and Dehghani, M. (2015). Combining distributed vector representations for words. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 95–101.

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

Lai, S., Liu, K., He, S., and Zhao, J. (2016). How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.

Lebret, R. and Collobert, R. (2013). Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*.

Lu, W., Chieu, H. L., and Löfgren, J. (2016). A general regularization framework for domain adaptation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 950–954.

McClosky, D. (2010). Any domain parsing: automatic domain adaptation for natural language parsing.

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

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Mnih, A. and Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273.

Patel, K., Patel, D., Golakiya, M., Bhattacharyya, P., and Birari, N. (2017). Adapting pre-trained word embeddings for use in medical coding. *BioNLP 2017*, pages 302–306.

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

Roy, A., Park, Y., and Pan, S. (2017). Learning domain-specific word embeddings from sparse cybersecurity texts. *arXiv preprint arXiv:1709.07470*.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Sarma, P. K., Liang, Y., and Sethares, W. A. (2018). Domain adapted word embeddings for improved sentiment classification. *arXiv preprint arXiv:1805.04576*.

Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Sørensen, T. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr.*, 5:1–34.

Yang, W., Lu, W., and Zheng, V. (2017). A simple regularization-based algorithm for learning cross-domain word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2898–2904.

Yin, W. and Schütze, H. (2015). Learning meta-embeddings by using ensembles of embedding sets. *arXiv preprint arXiv:1508.04257*.