DOMAIN ADAPTIVE EMBEDDINGS FOR TEXT

by

Prathusha Kameswara Sarma

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Electrical and Computer Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2019

Date of final oral examination: 05/09/2019

The dissertation is approved by the following members of the Final Oral Committee:
William A. Sethares, Professor, Electrical and Computer Engineering
Yingyu Liang, Assistant Professor, Department of Computer Science
Laurent Lessard, Assistant Professor, Electrical and Computer Engineering
Varun Jog, Assistant Professor, Electrical and Computer Engineering
Dhavan Shah, Professor, School of Journalism and Mass Communication

To amma, nana and santosh

ACKNOWLEDGMENTS

I consider myself immensely fortunate to have not one, but two remarkable families that supported me through this academic journey. My first and sincere heartfelt Thank You! is to my academic advisor Prof William (Bill) Sethares. Prof Sethares had faith and trust in my abilities when I had none, and supported me when I was at my lowest. He taught me that every incorrect hypothesis or result is a means to solidify my abilities to think scientifically and hence failure is just a prerequisite for success. Thank you for teaching me something this invaluable, I hope I am able to impart the same lessons in future to my colleagues.

My second Thank You! is to my second advisor Prof Yingyu Liang. Prof Liang has always been up to working on new problems and has often provided timely and sound advice; advice that often worked as the much needed final push towards success! I thank him from the bottom of my heart for all his time and energy. Both my advisors never fail to inspire in me the need to formulate cogent scientific arguments and to strive for scientific perfection. I thank them for all their time and energy and wish to make them proud of me in my future endeavors as a researcher.

My third Thank You! is to Prof Dhavan Shah, who gave me the security and resources to pursue this research. Thank you for taking a chance on me and for running wild with my ideas. I wish to imbibe from you the ability to apply ideas across disciplines to produce remarkable research!

This thesis will be incomplete, without thanking Zhongkai, Jo and Rachel and all my other friends and collaborators at CHESS and SMAD. You guys made my time here in Madison, so much more enjoyable. I wish that I have been a fairly decent teacher to Zhongkai and I wish him all the best for the future.

As I conclude thanking key member of my academic family, I now turn towards my other bigger family.

There are no sufficient words that I can use to thank my mom. My personal warrior, my mom has fought tirelessly by my side, first to keep me alive since the day she conceived me, to every second of my existence. Its my mom's dream for me that culminated in the form of this thesis. Always pushing me to aim higher and

strive for excellence, her lessons in discipline and dogged pursuit for excellence enabled me to complete this thesis. I love you mom and I only wish I make you as proud, as you make me. I owe every opportunity that has come my way to every sacrifice my father has made to enable me. A man who embodies the true meaning of love and who demonstrated the power of that very emotion, Thank You! nana, I wear your name with pride! I am truly honored and blessed to be the child of my parents, who provided me with every opportunity and advantage that I could get. I wish to be such a parent to my own children when the time comes.

I come from a family of strong women, one of whom is my sister. Thank You! Akka, for all your struggles have been lessons in how we must never back down from a fight, no matter how hard it gets.

It is very rare to find a best friend, a guide, a teacher, an inspiration and a supportive companion in one single person. I am blessed to have my husband wear all these caps with humility that is endearing. Thank You! Santosh, this would not have been possible without your support and cooperation. Thank you for putting our life on hold, so that I could pursue my personal dream and ambition. I can't wait to hold your hand and take steps into our future and I promise, henceforth I will always be there for you. I would also like to thank my parents-in-law for raising such a remarkable human being and for being comforting in time of need. Thank you! and to you too Vishal for being you!

As I conclude this note of thanks, I only wish to say that the words written on these pages fail to do justice to the gratitude I feel for each and every member of this big family that I have. Once again my humble Thank You! to each and every one of you.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis presents algorithms that learn word embeddings for text from low resource data sets that are characterized by atypical word semantics. Word embeddings are learned in order to perform sentiment classification tasks on low resource data sets. In this thesis, we introduce 'domain adaptive' word embeddings that combine local domain semantics along with generic language semantics by learning correlations between domain specific and generic word embeddings. Correlations are learned in three different ways i) using linear Canonical Correlation Analysis (CCA) that learns a linear subspace where correlations between two sets of variables are maximized, ii) using Kernel CCA (KCCA) where a non linear subspace is learned to maximize correlation and iii) a shallow neural network that learns projection weights during end-to-end training of a deeper neural network, when performing particular language tasks such as sentiment classification.

We study the performance of our proposed domain adaptive word embeddings in several binary and multi-class classification tasks on data sets that i) vary in size from small (1000 points) to medium (6000 points) and ii) drawn from distinct domains. We also show that domain adaptive embeddings perform well when measuring for language shift, i.e the changes in word associations following a 'viral' news/media event. This thesis also proposes learning word embeddings using supervision via the Supervised Word Embeddings for Sentiment Analysis (SWESA) algorithm. We also learn multi-view word embeddings that combine information from text, audio and video views of data in order to perform sentiment analysis on data.

1 INTRODUCTION

An important area of Natural Language Processing (NLP) research is word representation learning. By learning efficient representations/embeddings of words as points in vector space, one aims to quantify semantic relationships through simple mathematical equations. Since words are building blocks for sentences, long text documents etc, quality of the word embedding directly effects the quality of sentence and document representations.

Traditionally word embeddings were learned as functions of word frequencies in text documents. However, word frequency and/or word pair co-occurrences alone are not sufficient to learn high quality word embeddings. This is because words are polysemous and are heavily influenced by the context in which they occur. Furthermore, context plays a crucial role in predicting the sentiment of a document. For example consider the following pair of sentences *'Can't wait for the Christmas party this weekend!'*, and *'Anxious to attend the Christmas party this weekend!'*. Both sentences have a lot of common words and yet convey contrasting sentiment. The first sentence conveys a positive attitude towards the party, while the second conveys a more negative sentiment. Often the word 'party' is used in a positive context, like in first sentence more than the negative context of the second sentence. However, if one were to analyze a data set made up of a population of people detailing their efforts with maintaining sobriety and/or detailing their issues with social anxiety, there would be more instances of the word 'party' having a more negative sentiment that the generally associated positive sentiment of the word.

The current state-of-the-art handles context in word embedding algorithms, and learns word embeddings by conditioning on all words to the left and right of a given word. This is a neural network based algorithm and is trained on an extremely large data set (3000M points). Choice of training data set and computation power required to train this model are typical to neural network based architectures. Neural network architectures though highly efficient are also extremely data hungry. Empirically the state-of-the-art achieves tremendous success on several downstream

task such as Sentiment Analysis, Question Answering, Natural Language Inference (NLI) etc on several benchmark data sets.

However, despite being heavily optimized these algorithms have some practical limitations. Firstly, the choice of training data set for most algorithms is often a text corpus like Wikipedia, that contains a large volume of articles across several topics. Thus, the trained word embeddings capture generic semantic and syntactic use of language. This is a limitation when the trained word embeddings are to be applied on a target data set with 'specific' domain semantics. An example of this, is when one wants to design an algorithm to predict, in real-time the sentiment of a user posting on an on-line support group for addiction disorders. Secondly, computing resources to train such models are expensive and it is often labor intensive to re-train these models on more specific data sets. Particularly, when the target data set is only a few hundred thousands of points big, the size of the data set largely limits the performance of data hungry models.

To overcome these limitations, there is a need to 'adapt' word embeddings to the domains they are to be used in. Domain Adaptive word embeddings preserve the general language semantics that highly optimized embedding models learn, along with embedding the overall semantics of the application domain. Adaptive word embeddings perform particularly well on downstream tasks such as Sentiment Analysis on data sets from domains that are characterized by their small size and strong domain specific semantics.

The focus of this thesis is to describe algorithms that are used to learn 'Domain Adaptive' embeddings. Chapter 2 introduces an algorithm that learns domain adaptive embeddings by learning correlations between generic embeddings obtained from a highly optimized embedding algorithm and domain specific embeddings learned on a target data domain. Chapter 3 introduces a 'shallow' adaptive neural network that propagates word level domain adaptation onto sentence level embeddings when used in Sentiment Analysis tasks. Chapter 4 introduces an algorithm that learns polarized word embeddings through supervision. Chapter 5 briefly introduces multi-view embeddings that learn from multiple sources of text, audio and video data and Chapter 6 teases out future work and concludes this thesis.

The rest of this chapter details the evolution of word embedding algorithms and briefly describes the state-of-the-art in embedding algorithms.

1.1 Symbolic Representation of Words

The earliest representation of words from a vocabulary was symbolic (Turney et al. (2010)), i.e words were represented by atomic symbols. For a very long time NLP and IR communities represented words using one-hot encoding vectors. An one-hot vector is a $1\times$ N vector of zeros, with one non-zero element. Location of the non-zero element is unique to each word. While this representation is simplistic, it is not practical for several reasons. Firstly, dimension of the vectors, N, often scales linearly with the size of the vocabulary. This results in large vectors that quickly become cumbersome to handle for vocabularies consisting of more than a couple of hundreds of words. Secondly, simple semantic relationships such as similarity are not preserved. For example consider a vocabulary of two words, 'great' and 'excellent'. Often these words are used synonymously to describe a positive experience, outcome etc. Assume the following one-hot encoding representation for each of these two words.

$$\begin{aligned} \text{great} &= [0, 1] \\ \text{excellent} &= [1, 0] \end{aligned}$$

Any measure of similarity such as the dot-product or cosine similarity will result in an outcome that would pronounce the two words unrelated. Thus despite being similar in meaning both words appear semantically independent.

While several ad-hoc procedures can be implemented to overcome these shortcomings, algorithms utilizing one-hot encodings often end up learning more parameters than required. To overcome these difficulties, embedding techniques exploiting word co-occurrence statistics were developed.

1.2 Classic Vector Space Models

One-hot encodings are localistic representations for words and do not contain much information that can be shared across different data sets/applications. Words are often better understood by considering word co-occurrences i.e words occurring together in text. Early Vector Space Models (VSMs) make use of word co-occurrence statistics to learn word embeddings as points in a N-dimensional space with each dimension capturing some semantic relationship.

Bag-of-Words

One of the earliest and simplest VSMs represents text as a function of words, irrespective of word order or grammar (Turney et al. (2010)). For example given the following text documents,

- 'The rain in Spain falls mainly in the plain.'
- 'Madrid is the capital city of Spain.'
- 'Spanish omelets are frittatas.'
- 'Spanish is spoken in Spain.'

Disregarding stop words such as 'the,' 'is,', 'are,' and 'in,', a subset of unique words from these text documets is, 'spain,' 'spanish,' 'rain,' 'plain,' 'madrid,' 'capital,' 'omelets' and 'frittatas'. A 'bag' of words (BoW) can be constructed from the unique words, i.e a matrix with words along the columns and corresponding text documents along the rows is constructed. Each (i,j)th entry of this matrix corresponds to the number of times word j appeared in document i. Figure 1.1 shows the BoW matrix for the example documents described in this section. Each row represents the text document in vector space, while each column represents a word from the vocabulary in vector space. Unlike one-hot encoding, BoW word embeddings preserve some semantic relationships such as similarity. For example, calculating dot product between word vectors from the BoW matrix gives the following 'similar'

	capital	frittatas	madrid	omelets	plain	rain	spain	spanish
doc1	0	0	0	0	1	1	1	0
doc2	1	0	1	0	0	0	1	0
doc3	0	1	0	1	0	0	0	1
doc4	0	0	0	0	0	0	1	1

Figure 1.1: Bag of Words matrix of word counts for example documents.

word pairs (spain, spanish), (capital, madrid) to name among others. Dot product between document vectors will indicate that (doc1, doc2) and (doc3, doc4) are similar. Spurious relationships such as similarity between words (rain, spain) do exist, but this can be overcome.

Variants of this method include, dividing word counts of each word or the term frequency with the logarithm of the total number of texts divided by the number of texts containing a particular word. This measure is called term frequency-inverse document frequency (TF-IDF). Another popular metric used instead of raw word counts is the pointwise mutual information (PMI) or positive pointwise mutual information (PPMI). While the BoW framework enables measures for semantic similarity, the size of the word embeddings still scales proportional to the number of text documents. Another criticism to the BoW technique is that including statistics over all words in the text when learning a single word embedding may result in noisy word vectors.

Latent Semantic Analysis

To overcome issues with size, and noisy word co-occurrence statistics used to compute the TF-IDF or PPMI matrix in the BoW framework, an alternative technique was proposed. In this method, first, a raw counts/TF-IDF/PPMI matrix is constructed. Like in Figure 1.1 rows of the matrix represent documents while columns represent the unique words in the vocabulary. Next, singular value decomposition (SVD) of the matrix is performed. Word embeddings are obtained by projecting left singular vectors onto the k largest singular values. Similarly document embeddings

are obtained by projecting the right singular vectors onto the k largest singular values. This method is called Latent Semantic Analysis (LSA) (Dumais (2004), Deerwester et al. (1990)). The LSA framework finds low dimensional representation for word embeddings and is more robust to noise than the BoW framework. However, problems with this technique also have to do with the size of the vocabulary since computing the SVD can be expensive on large matrices.

1.3 Distributed Representation of Words

With the resurgence of neural network methods and the availability of large amounts of text data for analysis, several powerful word embedding algorithms that exploit distributional semantics of words in text are developed. The two most important algorithms from this class of algorithms are briefly described here.

Skip-gram with Negative Sampling

The Skip-gram with Negative Sampling (SGNS) algorithm (Mikolov et al. (2013a)), popularly known as word2vec, learns the embedding for a given word by observing words within a fixed size context (c) window around the given word. Consider words $w_1, w_2, \ldots, w_V$ from a vocabulary of size V. Let w_j be the center word, and consider a context of size $\alpha = 1$ such that we wish to calculate $\mathbb{P}(w_{j+\alpha}|w_j)$. For context of size $\alpha > 1$, and considering all words in the vocabulary, word embeddings can be obtained from the following optimization problem,

$$\min_{w_1,w_2,\ldots,w_V} \frac{1}{Vc} \sum_{j=1}^{V} \left[\sum_{|\alpha| \leq c} -\log(\mathbb{P}(w_{j+c}|w_j)) + \frac{1}{l} \sum_{k=1}^{l} \log(\mathbb{P}(w_k|w_j)) \right].$$

Note that the second term in the sum expresses a concept called 'Negative Sampling.' The aim of the SGNS objective is to maximize the probabilities of frequently occurring word combination as expressed by a word and its context while minimizing the probabilities of infrequent word pairs. Negative sampling indicates words pairs

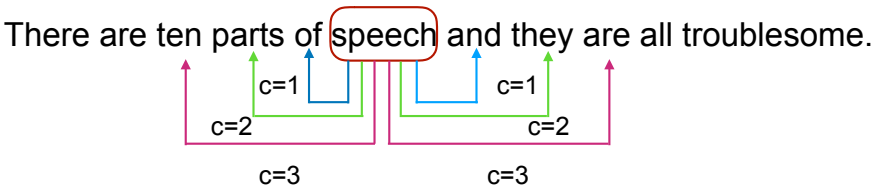

Figure 1.2: This figure illustrates the notion of a center word and a fixed context within the SGSN framework.

that appear infrequently in a given training corpus. For example, if the training set was a sample of documents obtained from IMDB words like 'Queen,"rock,' and 'band' are more likely to occur together within a context rather than the words 'Queen,' 'president' and 'bishop.' Negative sampling along with the Skip-gram objective aims to capture this difference. Figure 1.2 illustrates the concept of a center word and context. In this example, the center word is 'speech'. A context of size 1 include words to immediate left and right of the word 'speech,' while a context of size 2 includes the two words to the left and right of the word 'speech.'

With the power of neural network architectures and availability of large data sets such as the entire Wikipedia corpus for training, word2vec has demonstrated tremendous success in capturing semantic relationships such as analogies, similarities etc. Word embeddings obtained from word2vec when used in a simple optimization framework facilitate solving questions such as 'Man:King::Woman:?'. Owing to a number of desirable structural and semantic properties, embeddings obtained from word2vec are currently used to solve several semantic tasks such as analogies, relatedness etc ; while also being used to initialize several state-of-the-art algorithms for sentence encoding, sentiment analysis etc.

Global Vectors for Word Representation

The SGNS algorithm is an energy based model that exploits local word co-occurrence statistics to learn word embeddings. An alternative algorithm called Global Vectors for Word Representation (GloVe)(Pennington et al. (2014a)) is a matrix factorization

based approach that considers global word co-occurrence statistics to learn word embeddings. This particular algorithm exploits the properties of global log-bilinear regression models to learn low dimensional word embeddings. The GloVe objective is of the form,

$$\min_{w_1,w_2,...,w_V} \sum_{i,j=1}^{V} f(X_{i,j})(w_i^\top \tilde{w}_j + b_i + \tilde{b}_j - \log X_{i,j})^2.$$

In this objective, $\tilde{w}_j$ denotes the context word vector for the i^{th} word. $X_{i,j}$ is the co-occurrence term between words i and j and b_i and $\tilde{b}_j$ are bias terms. There are several functions that are suitable candidates for use when calculating $f(X_{i,j})$. Recent work on the theoretical aspects of GloVe and SGSN has established that under certain assumptions on the dimensions of word embeddings and when f is considered to be PPMI, embeddings learned from SGSN and GloVe are equivalent to embeddings learned via LSA of a PPMI word matrix.

1.4 Word embeddings via Language Models

While embedding algorithms like word2vec and GloVe learn high quality word embeddings by exploiting frequently occurring word pairs, often these algorithms encode the general linguistic semantics in the word embedding but do not capture word context information. Word context is crucial when accounting for polysemy in embeddings. By exploiting word co-occurrence alone, one may not capture the entire context in which a word occurs and thereby largely compromise on polysemy in representations. To overcome these limitations, word embeddings can be learned as parts of language models. The current state-of-the-art in embedding algorithms, ELMo and BERT learn word embeddings via language models.

Deep Contextualized Word Representations

Embeddings from Language Models (ELMo) (Peters et al. (2018b)), are word embeddings that are obtained from all the internal layers of a bidirectional language

model. Such embeddings are 'deep' since they are obtained as a function of the internal layers of a directional LSTM coupled with a language model objective. LSTMs (Hochreiter and Schmidhuber (1997)) are a special class of Recurrent Neural Networks (RNNs) that are capable of handling long-term dependencies. Empirically it is seen that the higher level states of the LSTMs capture context dependent aspects of word meaning, that is extremely useful for word sense disambiguation tasks. It is observed that the lower level states of the LSTMs capture syntax and are particularly useful for part-of-speech (POS) tagging tasks.

ELMo embeddings learn word embeddings as functions of the entire sentence and are computed on top of two bidirectional language models (biLM) with character convolutions as a linear function of the internal network states. A brief description of the language models and internal states is provided here.

Bi-directional Language Model

Given a sequence of N word tokens in a sentence $(w_1, w_2, \ldots, w_N)$, a forward language model computes the probability of a given token, conditioned on its history.

$$P(w_1, w_2, \ldots, w_N) = \prod_{k=1}^{N} P(w_k | w_1, w_2, \ldots, w_{k-1})$$

State-of-the-art in language models compute a context independent token representation $\mathbf{x}_k^{LM}$ either via token embeddings or a CNN over characters (A convolutional neural network (CNN) (Kim (2014a)) learns a single embedding from a matrix of word/token/character embedding by applying a convolution filter or kernel on the input word matrix. A pooling operation follows the convolution step to learn an aggregated feature vector.) and then pass it through L layers of forward LSTMs. At each kth position, each LSTM layer outputs a context dependent representation $\overrightarrow{\mathbf{h}}_{k,j}^{LM}$, where $j = 1, 2, \ldots, L$. The top layer LSTM output, $\overrightarrow{\mathbf{h}}_{k,L}^{LM}$ is used to predict the next word in the sentence w_{k+1} with a Softmax layer that applies a Softmax function to obtain normalized probability distributions over the words in the vocabulary.

A backward LM is similar to a forward LM with the exception that it computes

the sequence of reversed sentence tokens, i.e predict the previous word given its future context. The backward LM computes

$$P(w_1, w_2, \ldots, w_N) = \prod_{k=1}^{N} P(w_k | w_{k+1}, w_{k+2}, \ldots, w_N).$$

Implementing a backward LM is similar to a forward LM with each backward LSTM layer j in a L layer deep model producing representations $\overleftarrow{\mathbf{h}}_{k,j}^{LM}$ of w_k given $(w_{k+1}, w_{k+2}, \ldots, w_N)$.

As suggested by its name, a biLM combines both a forward LM and a backward LM to jointly maximize for the forward and backward loss function

$$\sum_{k=1}^{N} \Big(\log(P(w_k | w_1, w_2, \ldots, w_{k-1}); \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log(P(w_k | w_{k+1}, w_{k+2}, \ldots, w_N); \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big).$$

Parameters for token representation (Θ_x) and Softmax layer (Θ_s) are tied together and separate parameters for forward and backward LSTMs are learned.

ELMo embeddings are learned in a task specific manner. For each word token w_k, the biLM computes $2L + 1$ representations,

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} | j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} | j = 0, \ldots, L\} \end{aligned}$$

here $\mathbf{h}_{k,0}^{LM}$ is the initial word embedding layer and $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM} | \overleftarrow{\mathbf{h}}_{k,j}]$.

When used for downstream tasks, ELMo collapses all layers in R into a single vector $\mathbf{ELMo}_k = E(R_k; \Theta_e)$. Task specific embeddings are obtained by learning task specific weights for all the biLM layers as,

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

Pre-training of Deep Bidirectional Transformers for Language Understanding

Most language model based embedding approaches are either i) feature-based (ELMo) or ii) fine-tuning based. Fine-tuning based approaches Devlin et al. (2018), Radford et al. (2018) learn fewer parameters and largely focus on fine-tuning pre-trained model parameters on a specific downstream task. However, modeling limitations restrict the applicability of language models.

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. (2018)) overcomes limitations posed by most language model based embedding techniques by i) proposing a new language model objective based on 'masked language model' (MLM) task and ii) fusing of left and right contexts. A masked language model task involves randomly masking some tokens in the input sentence, and the objective is to predict the original word id based on the context alone.

At the heart of the BERT model is the Transformer block (Vaswani et al. (2017)). BERT makes use of a multi-layered bidirectional transformer encoder. Input to the BERT model architecture is represented by summing over word, segment and position embeddings. Particularly, the first token of every sequence is always the special classification embedding [CLS]. The final hidden state for this token is used to represent the aggregate sequence representation in classification tasks.

Since rather than feature based models like ELMo, BERT is a task based model, we briefly describe the two tasks used to pre-train the BERT encoder.

Masked Language Model

To train a deep bidirectional encoder, some percentage of the input word tokens are masked at random, and then the model predicts the masked tokens. Final hidden states corresponding to the masked token are fed into an output Softmax layer over the vocabulary. Unlike denoising auto-encoders, only masked words are predicted and the entire input is not reconstructed. Since the masked token is never seen during fine-tuning, there is a potential mismatch between data in pre-training and fine-tuning. To overcome this, not all word tokens are replaced with [MASK].

Instead, for a given input sentence e.g,'This is an interesting thesis document', 15% of word tokens in the input are randomly chosen e.g, the word 'interesting' and,

- 80% of the time replace the chosen word with [MASK], e.g 'This is an **interesting** thesis document' $\rightarrow$ 'This is an **[MASK]** thesis document'.
- 10% of the time replace the chosen word with any random word, e.g 'This is an **interesting** thesis document' $\rightarrow$ 'This is an **apple** thesis document'.
- 10% of the time keep the chosen word unchanged, i.e 'This is an **interesting** thesis document' $\rightarrow$ 'This is an **interesting** thesis document'.

When training the transformer encoder, the encoder does not know which words are to be predicted or which words are randomly replaced and so the encoder has to keep representations of every input token. Since at any time, only 15% of the tokens are predicted in each batch, it is likely that more pre-training steps may be required for the model to converge. However, authors of (Devlin et al. (2018)) demonstrate that despite the slower convergence of BERT compared to left-to-right conditioned models, empirical improvements of BERT far exceed the increased training cost.

Next Sentence Prediction

The next sentence prediction task based training of the biLM is useful particularly for applications in downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI). In order to train a biLM that understands relationships between two sentences, not captured by language modeling alone, a task involving sentence prediction in a framework similar to masked token prediction is set up. From an input corpus, when choosing training data of sentence **A** and **B** pairs, 50% of the time **B** is the actual next sentence that follows **A**, and 50% of the time it is a random sentence from the corpus. For example,

- – Input: [CLS] This research is [MASK] [SEP] Potential future work is [MASK]

  - Label: IsNext

- - Input: [CLS] This research is [MASK] [SEP] The weather in Madison is [MASK]
  - Label: NotNext

By pre-training on all of Wikipedia (2500M words) plus the BookCorpus (800M words) the authors demonstrate that BERT attains the state-of-the-art results in 9 out of 11 important NLP tasks such as Question Answering (QA), Natural Language Inference, Sentiment Analysis etc.

2 DOMAIN ADAPTED WORD EMBEDDINGS FOR IMPROVED SENTIMENT CLASSIFICATION

2.1 Overview

This chapter introduces the first of two methods proposed to learn Domain Adapted word embeddings. Motivation for this work comes from the influence of context words on individual word representations. The proposed algorithm in this chapter first appeared in out paper (K Sarma et al. (2018)) and empirically, comparisons are made with the then state-of-the art word embedding algorithms such as word2vec, GloVe and sentence level encoding algorithms such as Infersent.

2.2 Abstract

Generic word embeddings are trained on large-scale generic corpora; *Domain Specific* (DS) word embeddings are trained only on data from a domain of interest. This chapter introduces a method to combine the breadth of generic embeddings with the specificity of domain specific embeddings. The resulting embeddings, called *Domain Adapted* (DA) word embeddings, are formed by first aligning corresponding word vectors using Canonical Correlation Analysis (CCA) or the related nonlinear Kernel CCA (KCCA) and then combining them via convex optimization. Results from evaluation on sentiment classification tasks show that the DA embeddings substantially outperform both generic, DS embeddings when used as input features to standard or state-of-the-art sentence encoding algorithms for classification.

2.3 Introduction

Generic word embeddings such as Glove and word2vec (Pennington et al. (2014a); Mikolov et al. (2013c)) which are pre-trained on large bodies of raw text, have

demonstrated remarkable success when used as features for supervised learning problems. There are, however, many applications with domain specific vocabularies and relatively small amounts of data. The performance of generic word embeddings in such applications is limited, since word embeddings pre-trained on generic corpora do not capture domain specific semantics/knowledge, while embeddings learned on small data sets are of low quality.

A concrete example of a small-sized domain specific corpus is the Substances User Disorders (SUDs) data set(Quanbeck et al. (2014); Litvin et al. (2013)). This data set contains messages from discussion forums for people with substance addiction. Such forums are part of mobile health intervention treatments that encourages participants to engage in sobriety-related discussions. The goal of such treatments is to analyze content of participants' digital media content and provide human intervention via machine learning algorithms. This data is both domain specific and limited in size. Other examples of similar data sets include customer support tickets reporting issues with taxi-cab services, product reviews, reviews of restaurants and movies, discussions by special interest groups and political surveys. In general these data sets are common in domains where words have different sentiment from what they would have elsewhere.

These data sets present significant challenges for word embedding learning algorithms. First, words in data on specific topics have a different distribution than words from generic corpora. Hence using generic word embeddings obtained from algorithms trained on a corpus such as Wikipedia, would introduce considerable errors in performance metrics on specific downstream tasks such as sentiment classification. For example, in SUDs, discussions are focused on topics related to recovery and addiction; the sentiment behind the word 'party' may be very different in a dating context than in a substance abuse context. Thus domain specific vocabularies and word semantics may be a problem for pre-trained sentiment classification models (Blitzer et al. (2007)).

Second, there is insufficient data to completely retrain a new set of word embeddings. The SUD data set consists of a few hundred people and only a fraction of these are active (Firth et al. (2017), Naslund et al. (2015)). This results in a small

data set of text messages available for analysis. Furthermore, these messages are unstructured and the language used is informal.

Fine-tuning the generic word embedding also leads to noisy outputs due to the highly non-convex training objective and the small amount of the data. Since such data sets are common, a simple and effective method to adapt word embedding approaches is highly valuable. While existing work (e.g Yin and Schütze (2016)) combines word embeddings from different algorithms to improve upon intrinsic tasks such as similarities, analogies etc, there does not exist a concrete method to combine multiple embeddings for extrinsic tasks. In this chapter we propose a method for obtaining high quality domain adapted word embeddings that capture domain specific semantics and are suitable for tasks on the specific domain. Our contributions are as follows.

1. We propose an algorithm to obtain Domain Adapted (DA) embeddings. DA embeddings are obtained by performing three steps. (i) First, generic embeddings are obtained from algorithms such as Glove or word2vec that are trained large corpora (such as Wikipedia, common crawl). (ii) Next we learn domain specific (DS) embeddings by applying algorithms such as Latent Semantic Analysis (LSA) on the domain specific corpus. (iii) We then perform Canonical Correlation Analysis (CCA) or kernelized CCA (KCCA) to obtain projected DS and projected generic embeddings. The projected DS and generic embeddings are linearly combined via an optimization formulation to obtain a single DA embedding for each word.

2. We propose two optimization based approaches to combining generic and DS embeddings. In the first method, we minimize the sum of squared distance of the DA embeddings from the projected embeddings. The second approach combines projected embeddings in such a way that the document clusters are tightly packed. This helps in our downstream sentiment analysis task by separating out the clusters. This method is particularly successful on the most idiosyncratic A-CHESS data set (Table 2.4).

3. We demonstrate the efficacy of our embeddings by measuring the accuracy of our classifiers built using various embeddings on a sentiment analysis task. In the first set of experiments (Table (2.2)) we train logistic regression classifiers using a bag-of-words (BOW) framework, for the problem of sentiment analysis. Our experimental results show that the classifier built using DA word embeddings outperform the classifiers built using Glove, word2vec or LSA. Our classifier also outperforms the classifier built using the embeddings output by the concSVD algorithm (Yin and Schütze (2016)) which, obtains a word embedding by performing SVD on a matrix of word embeddings.

4. In the second set of experiments we demonstrate the efficiency of DA embeddings when used to initialize InferSent; a bi-LSTM, encoder/decoder architecture that learns sentence embeddings from input word embeddings. The resulting document embeddings are classified using logistic regression classifier. Performance metrics (see Table 2.3) show that DA embeddings outperform generic embeddings such as Glove common crawl, when used to initialize InferSent. Furthermore, we also outperform RNTN, which is a recursive neural network based sentiment analysis algorithm (Socher et al. (2013)).

2.4 Domain Adapted Word Embeddings

Training word embedding algorithms on small data sets leads to noisy outputs, due to lack of data, while embeddings from generic corpora fail to capture specific local meanings within the domain. For example, the word "alcohol" has a somewhat neutral to a mildly positive tone in the common crawl corpus, whereas this same word has a strong negative sentiment in a substance use disorder (SUD) dataset. In order to learn useful word embeddings that incorporate the sentimentality of a small target corpus, we propose learning domain specific embeddings obtained by applying word embedding algorithms on the given target corpus and generic embeddings, obtained using applying word embedding algorithms on large generic

corpora, using CCA or kernel CCA (KCCA).

Let $\mathbf{W}_{DS} \in \mathbb{R}^{|V_{DS}| \times d_1}$ be the matrix whose columns are the domain specific word embeddings (obtained by, e.g., the LSA algorithm on the domain specific data set), where V_{DS} is its vocabulary and d_1 is the dimension of the embeddings. Similarly, let $\mathbf{W}_G \in \mathbb{R}^{|V_G| \times d_2}$ be the matrix of generic word embeddings (obtained by, e.g., GloVe algorithm on the Common Crawl data), where V_G is the vocabulary and d_2 is the dimension of the embeddings. Let $V_\cap = V_{DS} \cap V_G$. Let $\mathbf{w}_{i,DS}$ be the domain specific embedding of the word $i \in V_\cap$, and $\mathbf{w}_{i,G}$ be its generic embedding. For one dimensional CCA, let ϕ_{DS} and ϕ_G be the projection directions of $\mathbf{w}_{i,DS}$ and $\mathbf{w}_{i,G}$ respectively. Then the projected values are,

$$\begin{aligned} \bar{w}_{i,DS} &= \mathbf{w}_{i,DS}\, \phi_{DS} \\ \bar{w}_{i,G} &= \mathbf{w}_{i,G}\, \phi_G. \end{aligned} \tag{2.1}$$

CCA maximizes the correlation ρ between $\bar{w}_{i,DS}$ and $\bar{w}_{i,G}$ to obtain ϕ_{DS} and ϕ_G such that

$$\rho(\phi_{DS}, \phi_G) = \max_{\phi_{DS}, \phi_G} \frac{\mathbb{E}[\bar{w}_{i,DS} \bar{w}_{i,G}]}{\sqrt{\mathbb{E}[\bar{w}_{i,DS}^2]\mathbb{E}[\bar{w}_{i,G}^2]}} \tag{2.2}$$

where the expectation is over all words $i \in V_\cap$.

The d-dimensional CCA with $d > 1$ can be defined recursively. Suppose the first $d-1$ pairs of canonical variables are defined. Then the d^{th} pair is defined by seeking vectors maximizing the same correlation function subject to the constraint that they be uncorrelated with the first $d-1$ pairs. Equivalently, matrices of projection vectors $\mathbf{\Phi}_{DS} \in \mathbb{R}^{d_1 \times d}$ and $\mathbf{\Phi}_G \in \mathbb{R}^{d_2 \times d}$ are obtained for all vectors in $\mathbf{W}_{DS}$ and $\mathbf{W}_G$ where $d \leq \min\{d_1, d_2\}$. Embeddings obtained by $\bar{\mathbf{w}}_{i,DS} = \mathbf{w}_{i,DS}\, \mathbf{\Phi}_{DS}$ and $\bar{\mathbf{w}}_{i,G} = \mathbf{w}_{i,G}\, \mathbf{\Phi}_G$ are projections along the directions of maximum correlation. The final domain adapted embedding for word i is given by $\hat{\mathbf{w}}_{i,DA} = \alpha \bar{\mathbf{w}}_{i,DS} + \beta \bar{\mathbf{w}}_{i,G}$. We next propose optimization based algorithms to determine α, β.

2.5 α, β that minimizes the sum of squared distances

One way to determine α and β is to find DA embeddings such that in the CCA transformed space, the new DA embeddings are as close as possible to both generic and DS embeddings. This is expressed by the following optimization problem,

$$\min_{\alpha,\beta} \|\bar{\mathbf{w}}_{i,DS} - (\alpha\bar{\mathbf{w}}_{i,DS} + \beta\bar{\mathbf{w}}_{i,G})\|_2^2 + \|\bar{\mathbf{w}}_{i,G} - (\alpha\bar{\mathbf{w}}_{i,DS} + \beta\bar{\mathbf{w}}_{i,G})\|_2^2. \tag{2.3}$$

Solving (2.3) gives $\alpha = \beta = \frac{1}{2}$, i.e., the new vector is equal to the average of the two projections:

$$\hat{\mathbf{w}}_{i,DA} = \frac{1}{2}\bar{\mathbf{w}}_{i,DS} + \frac{1}{2}\bar{\mathbf{w}}_{i,G}. \tag{2.4}$$

2.6 α, β to minimize the sum of cluster variance

A major goal of learning domain adapted embeddings is to use them in a downstream task such as sentiment analysis. To facilitate better sentiment analysis it helps if the cluster of positive and negative documents are tightly clustered. That is, we would like to minimize the sum of variance of each individual cluster of documents. This can be cast as a convex optimization problem that has a closed form solution as shown in the following theorem.

Theorem 2.1. *Let $\beta = 1 - \alpha$. Then, the optimal value of α that minimizes the sum of the variance of document clusters is given by the following set of equations*

$$\tilde{\alpha} = \frac{\frac{1}{k}\sum_{i=1}^{k}(d_{g_{p_i}} - \hat{\mu}_p)^\top(\bar{\mu}_p - \bar{d}_{p_i}) + \frac{1}{N-k}\sum_{i=1}^{N-k}(d_{g_{n_i}} - \hat{\mu}_n)^\top(\bar{\mu}_n - \bar{d}_{n_i})}{\frac{1}{k}\sum_{i=1}^{k}(\bar{\mu}_p - \bar{d}_{p_i})^\top(\bar{\mu}_p - \bar{d}_{p_i}) + \frac{1}{N-k}\sum_{i=1}^{N-k}(\bar{\mu}_n - \bar{d}_{n_i})^\top(\bar{\mu}_n - \bar{d}_{n_i})} \tag{2.5}$$

$$\alpha = \max(0, \min(\tilde{\alpha}, 1)). \tag{2.6}$$

Proof. Assume that a DA embedding is expressed as, $\hat{\mathbf{w}}_{i,DA} = \alpha\bar{\mathbf{w}}_{i,DS} + (1-\alpha)\bar{\mathbf{w}}_{i,G}$. Further, let each i^{th} document be expressed as the sum of constituent word embed-

dings,

$$d_i = \sum_{j=1}^{n} \hat{\mathbf{w}}_{j,DA} = \sum_{j=1}^{n} (\bar{\mathbf{w}}_{j,G} + \alpha(\bar{\mathbf{w}}_{j,DS} - \bar{\mathbf{w}}_{j,G})) = d_{g_i} + \alpha \bar{d}_i.$$

Suppose, there are N documents of which k are positive and $N-k$ are negative. Also, let μ_p, μ_n denote the cluster center of all positive and negative documents respectively. We can determine α that minimizes the sum of cluster variances by solving

$$\min_{\alpha \in [0,1]} \frac{1}{k} \sum_{i=1}^{k} ||d_{p_i} - \mu_p||_2^2 + \frac{1}{N-k} \sum_{i=1}^{N-k} ||d_{n_i} - \mu_n||_2^2 \tag{2.7}$$

Here μ_p and μ_n are centers of positive and negative document cluster centers. Taking means of clusters, we get $\mu_p = \frac{1}{k} \sum_{i=1}^{k} (d_{g_{p_i}} + \alpha \bar{d}_{p_i}) = \hat{\mu}_p + \alpha \bar{\mu}_p$. Similarly $\mu_n = \hat{\mu}_n + \alpha \bar{\mu}_n$.

$$\min_{\alpha \in [0,1]} \frac{1}{k} \sum_{i=1}^{k} ||(d_{g_{p_i}} - \hat{\mu}_p) - \alpha(\bar{\mu}_p - \bar{d}_{p_i})||_2^2 + \frac{1}{N-k} \sum_{i=1}^{N-k} ||(d_{g_{n_i}} - \hat{\mu}_n) - \alpha(\bar{\mu}_n - \bar{d}_{n_i})||_2^2. \tag{2.8}$$

The above problem is a very simple convex minimization problem. Differentiating w.r.t. α and setting it to 0, and projecting the resulting solution onto the interval $[0, 1]$ we get the desired result. □

2.7 Kernel CCA

Because of its linear structure, the CCA in (2.2) may not always capture the best relationships between the two matrices. To account for nonlinearities, a kernel function, which implicitly maps the data into a high dimensional feature space, can be applied. For example, given a vector $\mathbf{w} \in \mathbb{R}^d$, a kernel function K is written in the form of a feature map φ defined by $\varphi : \mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_d) \mapsto \varphi(\mathbf{w}) =$

$(\varphi_1(\mathbf{w}), \ldots, \varphi_m(\mathbf{w}))(d < m)$ such that given $\mathbf{w}_a$ and $\mathbf{w}_b$

$$K(\mathbf{w}_a, \mathbf{w}_b) = \langle \varphi(\mathbf{w}_a), \varphi(\mathbf{w}_b) \rangle.$$

In kernel CCA, data is first projected onto a high dimensional feature space before performing CCA. In this work the kernel function used is a Gaussian kernel, i.e.,

$$K(\mathbf{w}_a, \mathbf{w}_b) = \exp\Big(- \frac{|| \mathbf{w}_a - \mathbf{w}_b ||^2}{2\sigma^2} \Big).$$

The implementation of kernel CCA follows the standard algorithm described in several texts such as Hardoon et al. (2004); see reference for details.

2.8 Experimental Evaluation

In this section we evaluate DA embeddings in binary sentiment classification tasks on four standard data sets. Document embeddings are obtained via i) a standard bag-of-words framework, in which documents are expressed as the weighted combination of their constituent word embeddings and ii) by initializing a state-of-the-art-sentence encoding algorithm, InferSent (Conneau et al. (2017a)) with DA word embeddings to obtain sentence embeddings. Encoded sentences are then classified using a logistic regressor. Performance metrics reported are average precision, F-score and AUC. All hyperparameters are tuned via 10 fold cross validation.

Data Sets

Experiments are conducted using four data sets which differ in vocabulary and content. All four arise in specific domains and hence illustrate the objective of this work. The four data sets are:

- The **Yelp data set** consists of 1000 restaurant reviews obtained from Yelp. Each review is associated with a 'positive' or 'negative' label. There are a total

of 2049 distinct word tokens in this data set.

- The **Amazon data set** consists of 1000 product reviews with 'positive' or 'negative' labels obtained from Amazon. It has 1865 distinct tokens.
- The **IMDB data set** consists of 1000 movie reviews with binary 'positive' and 'negative' labels obtained from IMDB. It has 3075 distinct tokens.
- The **A-CHESS data set** is a proprietary data set[1] obtained from a study involving users with alcohol addiction. Text data is obtained from a discussion forum in the A-CHESS mobile app (Quanbeck et al. (2014)). There are a total of 2500 text messages, with 8% of the messages indicative of relapse risk. Since this data set is part of a clinical trial, an exact text message cannot be provided as an example. However, the following messages illustrate typical messages in this data set, *"I've been clean for about 7 months but even now I still feel like maybe I won't make it."* Such a message is marked as 'threat' by a human moderator. On the other hand there are other benign messages that are marked 'not threat' such as *"30 days sober and counting, I feel like I am getting my life back."* The aim is to eventually automate this process since human moderation involves considerable effort and time. This is an unbalanced data set (8% of the messages are marked 'threat') with a total of 3400 distinct work tokens.

The first three data sets are obtained from Kotzias et al. (2015).

Word embeddings, baselines and parameter settings

The following generic and DS word embeddings are used,

- **Generic word embeddings:** Generic word embeddings used are GloVe[2] from both Wikipedia and common crawl and the word2vec (Skip-gram) embeddings[3]. These generic embeddings will be denoted as Glv, GlvCC and w2v.

[1] Center for Health Enhancement System Services at UW-Madison
[2] https://nlp.stanford.edu/projects/glove/
[3] https://code.google.com/archive/p/word2vec/

- **DS word embeddings:** DS embeddings are obtained via Latent Semantic Analysis (LSA) and via retraining word2vec on the test data sets using the implementation in gensim[4]. DS embeddings via LSA are denoted by LSA and DS embeddings via word2vec are denoted by DSw2v. We also retrained GloVe on our test datasets to obtain domain specific word embeddings. Since, the performance of retrained Glove embeddings was similar to word2vec we shall not present the results of Glove based DS embeddings in this paper.
- **concatenation-SVD (concSVD) baseline:** Generic and DS embeddings are concatenated to form a single embeddings matrix. SVD is performed on this matrix and the resulting singular vectors are projected onto the d largest singular values to form word embeddings. The resultant word embeddings called meta-embeddings proposed by Yin and Schütze (2016) have demonstrated considerable success in intrinsic tasks such as similarities, analogies etc.

Dimensions of generic, DS and DA word embeddings are as follows,

Word embedding	Dimension
GloVe	100
word2vec	300
LSA	70
CCA-DA	68
KCCA-DA	68
GloVe common crawl	300
KCCA-DA(GlvCC)	300
concSVD	300

Table 2.1: This table presents the average dimensions of LSA, generic and DA word embeddings.

Kernel parameter estimation:

A rule-of-thumb for estimating the kernel parameter σ is to set σ equal to the median of pairwise distance between data points (Flaxman et al. (2016)). We use this rule to set the value of σ for all of our experiments that use kernel CCA.

[4] `https://radimrehurek.com/gensim/`

CCA is performed using the readily available installation in python. KCCA used in this work is implemented in python and follows closely the implementation developed by Bilenko and Gallant (2016).

Data Set		Embedding	Avg Precision	Avg F-score	Avg AUC
Yelp	$\mathbf{W}_{DA}$	KCCA(Glv, LSA)	85.36± 2.8	81.89±2.8	82.57±1.3
		CCA(Glv, LSA)	83.69± 4.7	79.48±2.4	80.33±2.9
		KCCA(w2v, LSA)	87.45± 1.2	83.36±1.2	84.10±0.9
		CCA(w2v, LSA)	84.52± 2.3	80.02±2.6	81.04±2.1
		KCCA(GlvCC, LSA)	**88.11± 3.0**	**85.35±2.7**	**85.80±2.4**
		CCA(GlvCC, LSA)	83.69± 3.5	78.99±4.2	80.03±3.7
		KCCA(w2v, DSw2v)	78.09± 1.7	76.04±1.7	76.66±1.5
		CCA(w2v, DSw2v)	86.22± 3.5	84.35±2.4	84.65±2.2
		concSVD(Glv, LSA)	80.14± 2.6	78.50±3.0	78.92±2.7
		concSVD(w2v, LSA)	85.11± 2.3	83.51±2.2	83.80±2.0
		concSVD(GlvCC, LSA)	84.20± 3.7	80.39±3.7	80.83±3.9
	$\mathbf{W}_{G}$	GloVe	77.13± 4.2	72.32±7.9	74.17±5.0
		GloVe-CC	82.10± 3.5	76.74±3.4	78.17±2.7
		word2vec	82.80± 3.5	78.28±3.5	79.35±3.1
	$\mathbf{W}_{DS}$	LSA	75.36± 5.4	71.17±4.3	72.57±4.3
		word2vec	73.08± 2.2	70.97±2.4	71.76±2.1
Amazon	$\mathbf{W}_{DA}$	KCCA(Glv, LSA)	86.30±1.9	83.00±2.9	83.39±3.2
		CCA(Glv, LSA)	84.68±2.4	82.27±2.2	82.78±1.7
		KCCA(w2v, LSA)	87.09±1.8	82.63±2.6	83.50±2.0
		CCA(w2v, LSA)	84.80±1.5	81.42±1.9	82.12±1.3
		KCCA(GlvCC, LSA)	**89.73±2.4**	85.47±2.4	85.56±2.6
		CCA(GlvCC, LSA)	85.67±2.3	83.83±2.3	84.21±2.1
		KCCA(w2v, DSw2v)	85.68±3.2	81.23±3.2	82.20±2.9
		CCA(w2v, DSw2v)	83.50±3.4	81.31±4.0	81.86±3.7
		concSVD(Glv, LSA)	82.36±2.0	81.30±3.5	81.51±2.5
		concSVD(w2v, LSA)	87.28±2.9	**86.17±2.5**	**86.42±2.0**
		concSVD(GlvCC, LSA)	84.93±1.6	77.81±2.3	79.52±1.7
	$\mathbf{W}_{G}$	GloVe	81.58±2.5	77.62±2.7	78.72±2.7
		GloVe-CC	79.91±2.7	81.63±2.8	81.46±2.6
		word2vec	84.55±1.9	80.52±2.5	81.45±2.0
	$\mathbf{W}_{DS}$	LSA	82.65±4.4	73.92±3.8	76.40±3.2
		word2vec	74.20±5.8	72.49±5.0	73.11±4.8

Table 2.2: This table shows results from the classification task on the Yelp and Amazon data sets using sentence embeddings obtained from weighted averaging of word embeddings. Metrics reported are average Precision, F-score and AUC and the corresponding standard deviations. Best performing embeddings and corresponding metrics are highlighted in boldface.

Data Set		Embedding	Avg Precision	Avg F-score	Avg AUC
IMDB	DA	KCCA(Glv, LSA)	73.84±1.3	73.07±3.6	73.17±2.4
		CCA(Glv, LSA)	73.35±2.0	73.00±3.2	73.06±2.0
		KCCA(w2v, LSA)	**82.36±4.4**	**78.95±2.7**	**79.66±2.6**
		CCA(w2v, LSA)	80.66±4.5	75.95±4.5	77.23±3.8
		KCCA(GlvCC, LSA)	54.50±2.5	54.42±2.9	53.91±2.0
		CCA(GlvCC, LSA)	54.08±2.0	53.03±3.5	54.90±2.1
		KCCA(w2v, DSw2v)	60.65±3.5	58.95±3.2	58.95±3.7
		CCA(w2v, DSw2v)	58.47±2.7	57.62±3.0	58.03±3.9
		concSVD(Glv, LSA)	73.25±3.7	74.55±3.2	73.02±4.7
		concSVD(w2v, LSA)	53.87±2.2	51.77±5.8	53.54±1.9
		concSVD(GlvCC, LSA)	78.28±3.2	77.67±3.7	74.55±2.9
	$\mathbf{W}_G$	GloVe	64.44±2.6	65.18±3.5	64.62±2.6
		GloVe-CC	50.53±1.8	62.39±3.5	49.96±2.3
		word2vec	78.92±3.7	74.88±3.1	75.60±2.4
	$\mathbf{W}_{DS}$	LSA	67.92±1.7	69.79±5.3	69.71±3.8
		word2vec	56.87±3.6	56.04±3.1	59.53±8.9
A-CHESS	DA	**KCCA(Glv, LSA)**	32.07±1.3	39.32±2.5	**65.96±1.3**
		CCA(Glv, LSA)	32.70±1.5	35.48±4.2	62.15±2.9
		KCCA(w2v, LSA)	33.45±1.3	**39.81±1.0**	65.92±0.6
		CCA(w2v, LSA)	33.06±3.2	34.02±1.1	60.91±0.9
		KCCA(GlvCC, LSA)	36.38±1.2	34.71±4.8	61.36±2.6
		CCA(GlvCC, LSA)	32.11±2.9	36.85±4.4	62.99±3.1
		KCCA(w2v, DSw2v)	25.59±1.2	28.27±3.1	57.25±1.7
		CCA(w2v, DSw2v)	24.88±1.4	29.17±3.1	57.76±2.0
		concSVD(Glv, LSA)	27.27±2.9	34.45±3.0	61.59±2.3
		concSVD(w2v, LSA)	29.84±2.3	36.32±3.3	62.94±1.1
		concSVD(GlvCC, LSA)	28.09±1.9	35.06±1.4	62.13±2.6
	$\mathbf{W}_G$	GloVe	30.82±2.0	33.67±3.4	60.80±2.3
		GloVe-CC	**38.13±0.8**	27.45±3.1	57.49±1.2
		word2vec	32.67±2.9	31.72±1.6	59.64±0.5
	$\mathbf{W}_{DS}$	LSA	27.42±1.6	34.38±2.3	61.56±1.9
		word2vec	24.48±0.8	27.97±3.7	57.08±2.5

Table 2.3: This table shows results from the classification task on the IMDB and A-CHESS data sets using sentence embeddings obtained from weighted averaging of word embeddings. Metrics reported are average Precision, F-score and AUC and the corresponding standard deviations. Best performing embeddings and corresponding metrics are highlighted in boldface.

Data Set	Embedding	Avg Precision	Avg F-score	Avg AUC
Yelp	GlvCC	86.47±1.9	83.51±2.6	83.83±2.2
	KCCA(GlvCC, LSA)	**91.06±0.8**	**88.66±2.4**	**88.76±2.4**
	CCA(GlvCC, LSA)	86.26±1.4	82.61±1.1	83.99±0.8
	concSVD(GlvCC,LSA)	85.53±2.1	84.90±1.7	84.96±1.5
	RNTN	83.11±1.1	-	-
Amazon	GlvCC	87.93±2.7	82.41±3.3	83.24±2.8
	KCCA(GlvCC, LSA)	**90.56±2.1**	**86.52±2.0**	**86.74±1.9**
	CCA(GlvCC, LSA)	87.12±2.6	83.18±2.2	83.78±2.1
	concSVD(GlvCC, LSA)	85.73±1.9	85.19±2.4	85.17±2.6
	RNTN	82.84±0.6	-	-
IMDB	GlvCC	54.02±3.2	53.03±5.2	53.01±2.0
	KCCA(GlvCC, LSA)	59.76±7.3	**53.26±6.1**	56.46±3.4
	CCA(GlvCC, LSA)	53.62±1.6	50.62±5.1	**58.75±3.7**
	concSVD(GlvCC, LSA)	52.75±2.3	53.05±6.0	53.54±2.5
	RNTN	**80.88±0.7**	-	-
A-CHESS	**GlvCC**	52.21±5.1	**55.26±5.6**	**74.28±3.6**
	KCCA(GlvCC, LSA)	**55.37±5.5**	50.67±5.0	69.89±3.1
	CCA(GlvCC, LSA)	54.34±3.6	48.76±2.9	68.78±2.4
	concSVD(GlvCC, LSA)	40.41±4.2	44.75±5.2	68.13±3.8
	RNTN	-	-	-

Table 2.4: This table shows results obtained by initializing InferSent encoder with different embeddings in the sentiment classification task. Metrics reported are average Precision, F-score and AUC along with the corresponding standard deviations. Best performing embeddings and corresponding metrics are highlighted in boldface We use $\alpha = 0.5$ for all of our experiments here.

Results from standard classification tasks

Tables 2.2 and 2.3 present results from the standard classification task on the Yelp and Amazon and IMDB and A-CHESS data sets respectively.. In this approach, we use a bag-of-words approach to combine word embeddings weighted by their term frequency counts. The resulting encoding $v = \gamma^\top \mathbf{W}$. Here $\gamma \in \mathbb{R}^{|V|}$ represents the weights for all the words in the sentence/document, and $\mathbf{W}$ is the matrix whose columns are word embeddings. A logistic regression classifier is then trained on the training data and used to predict the sentiment labels on the test data sets. From this table, it can be inferred that DA embeddings obtained by applying KCCA on GlvCC generic and LSA DS embeddings provide the best performing results on all data sets. Note that in these experiments $\alpha = \frac{1}{2}$ (2.5). On the Amazon data set,

concSVD achieves slightly better average F-score (86.17) and average AUC (86.42) over average F-score (85.47) and average AUC (85.56) obtained by KCCA (GlvCC, LSA). However, KCCA (GlvCC, LSA) achieves an average precision of 89.73 while concSVD achieves an average precision of 87.28. On the A-CHESS data set, owing to the imbalance in the classes, the best performing embedding is one that achieves maximum precision. From the table we can determine that KCCA (GlvCC, LSA) achieves the highest average precision of 36.38.

Results from InferSent encoding for classification

In this section DA embeddings are used to initialize a state-of-the-art sentence encoding algorithm, InferSent. The resultant sentence embeddings are then classified using a logistic regression classifier. Table 2.4 presents results from classifying sentences obtained from InferSent. First, the pre-trained encoder[5] initialized with GloVe common crawl embeddings is used to obtain vector representations of the input data. Next, InferSent is fine-tuned with a combination of GloVe common crawl embeddings and DA embeddings. DA embeddings are only obtained for a small subset of a vocabulary, so the combination is obtained by using the common crawl embeddings for the rest of the vocabulary. The same procedure is repeated with concSVD embeddings. Additionally, embeddings are compared against a classic sentiment classification algorithm, the Recursive Neural Tensor Network (RNTN) (Socher et al. (2013)). This is a dependency parser based sentiment analysis algorithm. Since the focus of this work is not on sentiment analysis algorithms per se, but on domain adaptation of word embeddings for extrinsic tasks, this is used as a baseline for comparison. From table 2.4 it can be inferred that KCCA(GlvCC, LSA) embeddings perform better than all other baselines for Yelp, Amazon and A-CHESS data sets. On the IMDB data set, RNTN performs best. This could be a case of (GlvCC, LSA) being bad initial guess embeddings for the IMDB data set. Performance of GlvCC embeddings from table 2.2 further support this conjecture.

[5] `https://github.com/facebookresearch/InferSent`

Also, InferSent produces superior sentence embeddings than simple averaging hence results from table 2.4 are better than results in tables 2.2 and 2.3.

Results from using α that minimizes the sum of cluster variances

As described in Theorem (2.1), α can be selected to minimizes variance of document clusters when learning DA embeddings. Since from tables 2.2, 2.3 and 2.4 we see that the best performing DA embedding is obtained by KCCA, results for this embedding alone are presented in table 2.5. Furthermore, empirically we did not observe much difference in CCA DA embeddings obtained using $\alpha = 0.5$ and α that minimizes the sum of cluster variances. From tables 2.3 and 2.4 observe that on the Yelp, Amazon and IMDB data sets, there is not much of difference in performance metrics for $\alpha = 0.5$ and the α obtained from Theorem (2.1). However, on the A-CHESS data set, α as obtained from Theorem (2.1) does better than $\alpha = 0.5$. This result is not surprising given that the word sentiments on the A-CHESS data set is highly atypical. This supports our hypothesis that using only generic embeddings such as the GloVe common crawl is not sufficient when analyzing datasets such as the A-CHESS dataset.

2.9 Discussion and Conclusion

In this chapter DA embeddings are obtained by optimizing a combination of generic and DS embeddings that are projected along directions of maximum correlation. The resulting DA embeddings are evaluated on sentiment classification tasks from four different data sets. Results show that while actual performance metrics vary from database to database, the optimized DA embeddings outperform both the generic and the DS word embeddings in a standard classification framework; as well as outperform concatenation based combination embeddings. This is a positive results since CCA/KCCA provides a principled formulation for combining multiple embeddings. In contrast, concatenating embeddings followed by SVD is an ad-hoc procedure and does not exploit correlations among multiple embeddings.

Data Set	Embedding	α	Avg Precision	Avg F-score	Avg AUC
Yelp	KCCA(Glv, LSA)	0.25	84.75±2.2	80.02±2.5	81.13±2.0
	KCCA(w2v, LSA)	0.45	87.74±2.2	83.57±2.6	84.27±2.4
	KCCA(GlvCC, LSA)	0.6	88.84±2.3	85.36±2.3	85.93±2.0
Amazon	KCCA(Glv, LSA)	0.35	85.63±1.3	84.64±1.9	84.84±1.6
	KCCA(w2v, LSA)	0.54	87.15±2.0	84.27±1.9	84.79±1.6
	KCCA(GlvCC, LSA)	0.4	90.42±2.2	87.48±2.3	87.92±2.0
IMDB	KCCA(Glv, LSA)	0.35	72.10±1.8	72.63±2.3	73.01±2.1
	KCCA(w2v, LSA)	0.4	83.01±1.6	79.10±1.2	79.96±2.0
	KCCA(GlvCC, LSA)	0.45	58.56±1.8	53.29±1.7	60.56±1.9
A-CHESS	KCCA(Glv, LSA)	0.4	37.32±1.6	41.64±2.8	66.13±2.1
	KCCA(w2v, LSA)	0.55	35.06±0.9	43.44±1.4	68.60±1.3
	KCCA(GlvCC, LSA)	0.75	38.65±3.1	43.03±2.2	67.26±2.2

Table 2.5: This table shows results using KCCA DA embeddings within a BoW framework. Since from tables 1 and 2 we see that the best performing DA embedding is obtained by KCCA, results for this embedding alone are presented in this table. α used minimizes the sum of cluster variances as shown in Theorem (2.1). Note that on the A-CHESS dataset the value of α is large. This observation supports our hypothesis that on domain specific data sets such as A-CHESS, using only generic embeddings such as the GloVe common crawl, as features for classification or to initialize algorithms such as InferSent is not sufficient.

The need for such DA embeddings is motivated by the limitations of performance of generic embeddings on data sets such as A-CHESS. Initializing InferSent with DA embeddings further improves the output from InferSent. This is encouraging because several NLP tasks such as Sentiment Analysis, POS tagging, etc., use algorithms that must be initialized with word embeddings. Initializing such algorithms with embeddings customized to a particular domain or data set will improve performance of these algorithms.

The next chapter proposes a solution that implements a shallow CNN based architecture to learn weights α and β that optimizes DA embeddings for particular downstream tasks such as sentiment classification, question-answer queries etc.

3 DOMAIN ADAPTIVE EMBEDDINGS FOR NATURAL LANGUAGE PROCESSING

3.1 Overview

In this chapter we continue learning Domain Adaptive embeddings using neural network based algorithms. The proposed algorithm learns embeddings in a task oriented manner using 'shallow' networks that perform well on limited data domains. Empirical results presented in this chapter extend to multiclass data sets. Performance of the proposed algorithm is compared against deep language model based embedding algorithms such as BERT (Devlin et al. (2018)).

3.2 Abstract

This chapter proposes a way to improve the performance of existing algorithms for text classification in domains with strong language semantics. A proposed domain adaptation layer learns weights to combine a generic and a domain specific (DS) word embedding into a domain adapted (DA) embedding. The DA word embeddings are then used as inputs to a generic encoder + classifier framework to perform a downstream task such as classification. This adaptation layer is particularly suited to data sets that are modest in size, and which are, therefore, not ideal candidates for (re)training a deep neural network architecture. Results on classification tasks using popular encoder architectures, including current state-of-the-art methods (with and without the shallow adaptation layer) show the effectiveness of the proposed approach.

3.3 Introduction

Domain Adaptation (DA) algorithms are becoming increasingly relevant in addressing issues related to i) lack of availability of training data in domains of interest and

in ii) exploiting domain idiosyncrasies to improve performance of out-of-domain algorithms. While some state of the art DA algorithms focus on improving on a downstream task, such as classification in bilingual or crosslingual framework (Hangya et al. (2018)), (Liu et al. (2018)), others (Zeng et al. (2018)), (Shang et al. (2018)) focus on tackling DA at the word level for various downstream applications such as classification, tagging etc.

Particularly, work by (An et al. (2018)) and (K Sarma et al. (2018)) address the issue of context-based domain adaptation within a single language. The authors of both works argue that word context causes significant changes in document level sentiment associations. The idea is illustrated by providing examples of words such as 'kill,' that has a more positive sentiment in a document describing video games than when used in a news article. Similarly, words used in social media and medical domains tend to be atypical and idiosyncratic as compared to generic language use.

Common evaluation metrics that can be used to assess the usefulness of DA algorithms are i) the performance of the DA algorithm on downstream classification tasks such sentiment analysis, where given a document, the sentiment label for the document is to be determined and ii) word level similarities determined through annotated gold standard data. Since adaptation can be viewed as a shift in the position of words within the embedding spaces across the domains of interest, shifts in the spatial position of the words can be measured to verify that the algorithm is indeed capturing domain knowledge. Once it is established that a given DA method is capturing domain semantics at the word level, the DA word embeddings can then be used as input to an encoder+task specific layer for applications in various downstream tasks.

The proposed algorithm applies a generic adaptation framework to tasks that make use of domain semantics. Contributions of this method are fourfold,

- The proposed generic domain adaptation layer can be interfaced with any neural network block with the aim of improving performance on a downstream task.

- We measure the significance of the shift in word position when represented in a generic embedding space and when represented in the adapted embedding space. The strategy is to construct DA word embeddings using the procedure described in the previous chapter (K Sarma et al. (2018)) and make use of a seed lexicon as in (Hangya et al. (2018)) and (Mikolov et al. (2013b)). However, rather than use the seed lexicon to transform words from two different domains into a common space, we use the seed lexicon to verify that the DA word embeddings have indeed captured significant domain semantics.

- We test our hypothesis on a recently introduced (Friedland et al. (2017))[1] data set of tweets from Liberal and Conservative Twitter users from the state of Wisconsin. As illustrated in (An et al. (2018)), political discourse makes for an interesting setting to study domain adaptation.

- We peform experiments on multiple binary and multiclass labeled data sets, to show that using a DA layer helps improve the performance of standard architectures on sentiment analysis tasks by $2 - 8\%$. We also, show that our DA architectures outperform sophisticated architectures such as BERT, LR-Bi-LSTM, Self-attention by $1 - 2\%$.

The rest of this chapter is organized as follows, Section 5.4 discusses related work. Section 3.5 describes in detail, the proposed algorithm, Section 5.6 presents the experimental results and Section 5.7 concludes this work.

3.4 Related Work

Central themes of this chapter tie into word level domain semantics, while being related to the overall objective of domain adaptation. In this section, we briefly discuss work that is close to the central theme of our proposed approach.

Recent work such as SEMAXIS (An et al. (2018)) investigates the use of word level domain semantics for applications beyond sentiment analysis. The authors

[1] https://github.com/naacl18sublong/Friedland

introduce the concept of a semantic axis based on word antonym pairs to capture semantic differences across corpora. Similarly, work by (Hamilton et al. (2016)) captures domain semantics in the form of sentiment lexicons via graph propagation. While both these lexical based approaches are similar to the ideas of this paper, a major difference is that like (K Sarma et al. (2018)), we do not make use of any predefined domain specific lexicons to capture domain semantics. Our idea is to use word occurrences and contexts to provide a raw estimate of the domain semantics. Using generic embedding spaces as baselines, adaptation is performed by projecting generic embeddings into a learned 'adaptation' space.

Typical downstream applications such as cross lingual and/or multi-domain sentiment classification, using algorithms proposed by (Hangya et al. (2018)), (Liu et al. (2018)), make use of DNNs with RNN blocks such as BiLSTMs to learn both generic and domain specific representations. Particularly, work focused on multi-domain sentiment classification as in (Liu et al. (2016)), (Nam and Han (2016)), proposes building neural architectures for each domain of interest, in addition to shared representation layers across all domains. While these techniques are effective, they are not ideal in domains with limited data.

On the other hand work such as ELMo (Peters et al. (2018a)) and BERT (Devlin et al. (2018)) propose deeply connected layers to learn sentence embeddings by exploiting bi-directional contexts. While both methods have achieved tremendous success in producing word (ELMo) and sentence (BERT) level encodings that perform well in several disparate NLP tasks such as question-answer solving, paraphrasing, POS tagging, sentiment analysis etc, these models are computationally expensive and require large amounts of training data. Particularly when used in a transfer learning setting, both algorithms assume that a large amount of data is present in the source as well as the target domains. In contrast, our proposed adaptation layer is particularly well suited in applications with limited data in the target domain.

Our proposed algorithms depart from these approaches by capturing domain semantics through shallow layers for use with generic encoder architectures. Since some of the most successful algorithms in text classification (Kim (2014b)) and

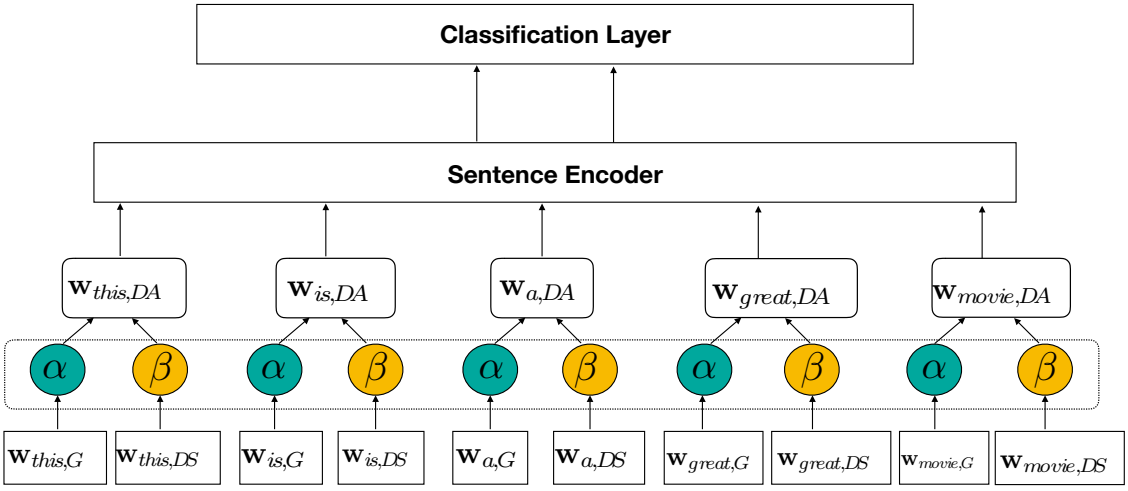

Figure 3.1: This figure illustrates the three part neural network model. The first part is an adaptation layer. The second part is a generic sentence encoder and the last part is a classification layer. Inputs to the shallow adaptation layer are the generic and the domain specific (DS) word embeddings. Output of the adaptation layer is the domain adapted (DA) word embedding.

sentence embeddings (Conneau et al. (2017b)) make use of CNN and BiLSTM building blocks, we suggest a generic adaptation framework that can be interfaced with these standard neural network blocks to improve performance on downstream tasks, particularly on small sized data sets.

3.5 Shallow Domain Adaptation

This section introduces a 'shallow' adaptation layer[2], to be interfaced with a neural network based sentence encoding layer, followed by a classification layer. Figure 3.1 describes a generic framework of the proposed model. Brief descriptions of the three layers follow.

[2]Codes for shallow adaptation will be made available upon acceptance of paper for publication.

Adaptation Layer: Generic word embeddings from GloVe (Pennington et al. (2014b)) or word2vec (Mikolov et al. (2013c)) are combined with an LSA-based (Deerwester et al. (1990)) Domain Specific (DS) word embedding using KCCA as proposed in (K Sarma et al. (2018)). Once the KCCA projections $(\bar{\mathbf{w}}_{i,G}, \bar{\mathbf{w}}_{i,DS})$ for the generic and DS word embeddings are obtained for a given word i, weights α and β are learned such that the domain adapted word embedding is $\bar{\mathbf{w}}_{i,DA} = \alpha\bar{\mathbf{w}}_{i,G} + \beta\bar{\mathbf{w}}_{i,DS}$. Weights α and β are learned by a single CNN layer. Since this layer learns only the weights used to obtain the DA embeddings, we call this a 'shallow' adaptation layer.

The output of this layer (i.e., the DA embeddings), is used to initialize a generic sentence encoder. Output from the sentence encoder is sent to a classification layer and the classification error is back propagated to update weights of the adaptation layer. By keeping the sentence encoder fixed, (i.e., the parameters of the encoder are not updated during back propagation), this method can be generalized for use with any pre-trained sentence encoder. This is advantageous, particularly in data sets that are too small to be trained end-to-end. Another advantage is that adaptation is not performed on fully connected layers that need to be trained end-to-end. This considerably reduces the number of parameters that need to be learned.

Figure **??** illustrates the CNN encoder architecture and the Adaptation Layer. Input to the adaptation layer is a word embedding of dimension $2d$, where d is the dimension of the generic and DS embeddings. For a given word, the generic and DS embeddings are concatenated and interleaved. A single layer CNN learns α and β via a 2×1 kernel. The domain adapted word $\mathbf{w}_{i,DA}$ is then passed as input to the CNN encoder as shown in the figure. Note that we also use this framework with a BiLSTM sentence encoder as in (Conneau et al. (2017b)) and is illustrated in Figures 3.2 and 3.3. Output of the BiLSTM units is max-pooled to obtain the sentence encoding.

Sentence Encoder: In this work the CNN encoders used is that of (Kim (2014b)) and the BiLSTM encoder used is that of (Conneau et al. (2017b)). Since both models have been extensively discussed in literature we shall skip elaborating on the basic model architectures. We reproduced the basic encoder models for the experiments

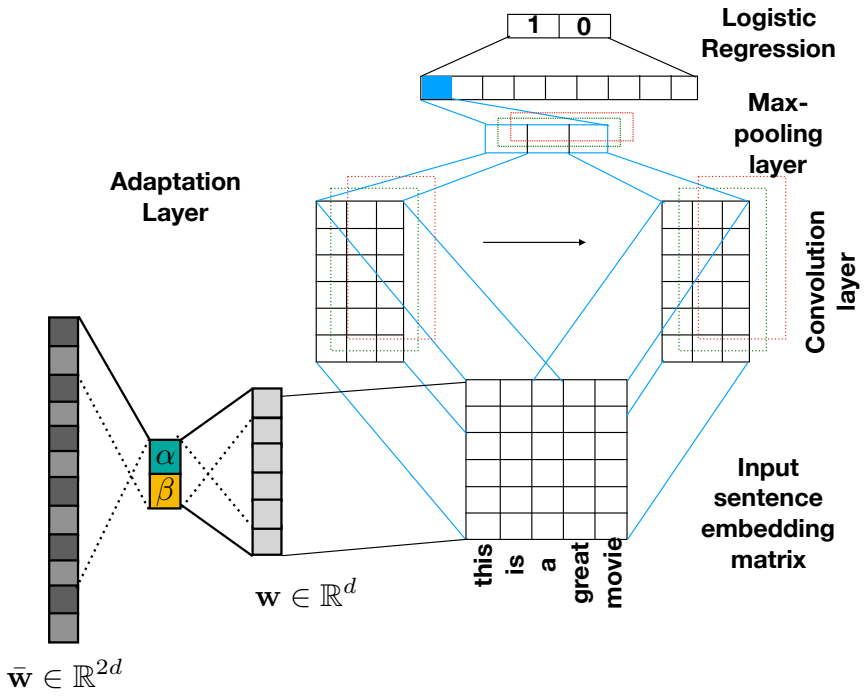

Figure 3.2: This figure illustrates the adaptation layer, that takes as input a $2d$ word embedding containing the generic and DS KCCA projections and learns a d dimensional DA word embedding. This DA word embedding is then input to the CNN encoder. The entire network can be trained end-to-end to learn parameters α and β in addition to the sentence embedding and classifier, or the network can be optimized to learn only the weights α and β and weights of the classifier.

in Section 5.6 with exception of few hyperparameters, since the choice of data sets in this work differs from that of (Kim (2014b)) and (Conneau et al. (2017b)).

Classification Layer: This layer learns weights for multi-class classification using a soft-max function.

Evaluating KCCA for Domain Adaptation

Experiments in Section 3.6 show that using the shallow adaptation layer along with generic sentence encoders improves the performance of the encoders on downstream classification tasks. However, in order to verify that the word level

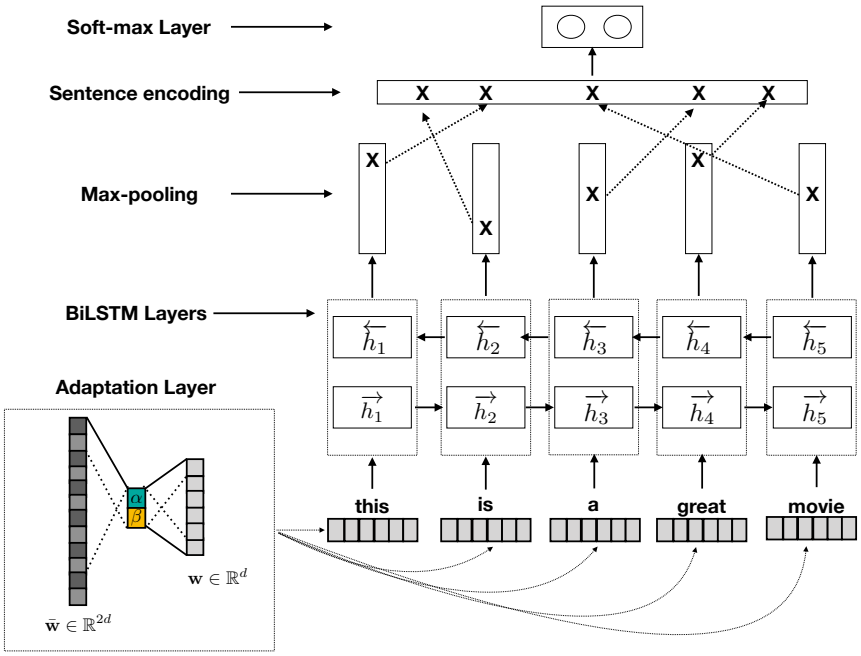

Figure 3.3: This figure illustrates the adaptation layer, that takes as input a $2d$ word embedding containing the generic and DS KCCA projections and learns a d dimensional DA word embedding. This DA word embedding is then input to the BiLSTM+max-pooling encoder. The entire network can be trained end-to-end to learn parameters α and β in addition to the sentence embedding and classifier, or the network can be optimized to learn only the weights α and β and weights of the classifier.

adaptation performed by the KCCA step can capture relevant domain semantics, we perform the following experiment.

Start with a dataset where language use is polarized. For example, the language used in the tweets of liberals likely differs from the language used in the tweets of conservatives as they express opinions on key political issues such as government, immigration, border control etc. These two vocabularies (Liberal and Conservative) are encoded in DS embeddings which are each paired with a generic embedding, and then again mapped into a common DA space. The words in this common space that are most different are then identified, and compared to a ground truth/gold

standard list of keywords which represent ideas or concepts on which the two groups of users are known to have the most polarized opinions. Comparing the gold standard list with the list derived from the DA embedding demonstrates the efficacy of the method. This experiment adopts the list of politically contentious words that appear in (Li et al. (2017)).

The remainder of this section provides details on this experiment. From the tokenized tweets of the users, construct a vocabulary of words V_{Lib} and V_{Con} that represent the Liberal and Conservative tweets respectively. Domain specific (DS) word embeddings are first constructed from V_{Lib} and V_{Con}. Then, KCCA projections are obtained for generic GloVe embeddings and the DS word embeddings for each user group, in order to construct DA word embeddings for Liberal $\hat{\mathbf{W}}_{DA(Lib)}$ and Conservative $\hat{\mathbf{W}}_{DA(Con)}$ users.

Next consider the set of common words in the two vocabularies, i.e $V_{common} = V_{Lib} \cap V_{Con}$. For analysis, perform a second KCCA to bring DA words in V_{common} from both Liberal V_{Lib} and Conservative V_{Con} vocabularies into a common subspace. To measure the 'shift' ψ, calculate the l_2 distance for each word embedding $\hat{\mathbf{W}}_i \in V_{common}$, i.e

$$\psi = ||\hat{\mathbf{W}}_{i,DA(Lib)} - \hat{\mathbf{W}}_{i,DA(Con)}||_2 \quad \forall i \in V_{common}$$

Words with large values of ψ are considered to have shifted the most between the domains of liberal and conservative users. Words are then ordered based on the magnitude of ψ, and cross referenced with words in the gold standard list.

In order to demonstrate that the KCCA can capture domain semantics when learning DA word embeddings, the following section presents an analysis against a random baseline.

Evaluations Against a Random Baseline

A random baseline is one that picks k words from V_{common}, randomly, and reports these k words as having shifted the most with respect to word use within tweets that correspond to Liberal and Conservative users. This random baseline follows

a hypergeometric distribution, which is a discrete probability distribution that calculates the probability of obtaining k successes (of a sample with a specific feature) from n draws, from a finite population of V, without replacement, where exactly K samples contain the specific feature.

For example, suppose $|V_{common}| = V$, and say the number of words in the gold standard list are K. Then the probability of the random baseline picking $k < K$ words out of n words is

$$Pr(X = k) = \frac{\binom{K}{k}\binom{V-K}{n-k}}{\binom{V}{n}}.$$

To calculate these probabilities for the KCCA-DA word embeddings, we shall use the LibCon data set of Section 3.5. A gold standard list of 136 key topics/concepts important to Liberals and Conservatives is obtained from the study by Li et al. (2017), and additional details about the LibCon data set and the gold standard list can be found in Section 3.6.

From the LibCon data set $|V_{common}| = V = 1573$. Out of these 1573 words, there are 74 words that are present in the gold standard list. Hence $K = 74$. After obtaining the DA word embeddings for words in V_{common} and calculating the shift ψ, we take the top 200 words that shifted most. From these words, at least 20 are present in the gold standard list.

The number of words-of-interest in a sample of 200 words forms a hypergeometric distribution with mean $\mu = n\frac{K}{V} \approx 9.4$, and standard deviation 2.7984.

Direct calculation shows that the probability of picking exactly 20 words from this subset is $Pr(X = 20) = 0.000346$. Similarly, the probability that 20 or more would have been chosen by chance is

$$p = \sum_{k=20}^{200} Pr(X = k)$$

which, by numerical calculation, is $p = 0.000524$. Such a result is highly unlikely to occur by chance. Accordingly, the KCCA domain adapted word embeddings are

indeed capturing something significant about the language usage in the domain.

3.6 Experimental Results

The discussion in Section 3.5 provides motivation for adopting the KCCA projections as a tool to calculate DA word embeddings. These DA word embeddings can be combined with standard neural network models to obtain sentence embeddings that can be used to improve performance on downstream tasks such as sentiment analysis and classification. To test the effectiveness of our proposed approach, we conduct a series of binary sentiment analysis/classification tasks on three datasets, namely LibCon, MMR, SST, using our adaptation layer on top of several standard architectures. Additionally, we also perform multiclass classification on the Beauty, Book and Music data sets obtained from (He et al. (2018a)). The encoders used in this paper are architectures that are common in the current SOTA for sentence encoders (Conneau et al. (2017b)) and multi-domain sentiment analysis (Liu et al. (2018)). These architectures are a BiLSTM based architecture followed by some sort of pooling and then a soft-max classification layer.

Our main result is that using the proposed domain adaptation layer on top of existing architectures helps improve the performance of the architecture by about $2 - 8\%$. Furthermore, we show that basic architectures enhanced with our DA layer outperform SOTA architectures built specifically for domain adaptation and transfer learning problems. We now give a description of the various datasets that we use and the various baseline algorithm used for our experimental comparisons.

Data Sets

Motivated by the fact that our DA layer is most useful when the target dataset is of modest size, we present our experimental results on three such binary sentiment datasets and three multi sentiment data sets. Furthermore, the text used in these datasets often uses the same set of words to express clearly contrastive sentiments, thereby setting up groups or domains of distinct word use within a single data set.

LibCon: The LibCon (**Lib**eral and **Con**servative) data set results from a study aimed at understanding, analyzing and modeling the changing landscape of political discourse of an entire state over a duration of 10 years (Friedland et al. (2017)). A key aspect of this study is the use of Twitter data to analyze the latent network structure among key players of Wisconsin's political-media ecology. To do this, data was drawn from a collection of Twitter data housed at the UW-Madison School of Journalism and Mass Communication.

Data is drawn from a 10% sample of Twitter messages worldwide from Twitter's API (Twitter 2012). Note that the period of data collection corresponds to the re-election efforts of Governor Walker (of Wisconsin). Snowball sampling is employed to identify important Twitter handles active in four 28-day periods in the first half of 2012. A hand curated list of Twitter handles corresponding to key players in Wisconsin's political climate is identified. Combined with Twitter handles of nationwide users that interact frequently with users in this list, the final list of Twitter handles of interest is formed.

A graph with Twitter handles as nodes and edges representing a re-tweet relationship between Twitter handles is formed. There are a total of $13,885$ nodes and $23,802$ edges in this graph. Handles are confirmed to their corresponding political associations by verifying their party membership. The Twitter handle pool includes a balanced representation from Republican, Democratic, pro-Walker and anti-Walker accounts. In depth description of data collection methodologies and outcomes can be found in (Friedland et al. (2017)) and readers should consult this source for additional information on this data set. A subsample of 1000 tweets, each from known Liberal (Democratic party) and Conservative (Republican party) users are considered for experiments here.

Gold standard list of topics key to both Democrat and Republican parties are obtained by analyzing primary debate data over a period of almost 20 years (1999-present) (Li et al. (2017)). A combination of modeling techniques involving semantic spaces and neural network architectures is used to arrive at the final list of 136 top/key concepts such as 'Army', 'Border', 'Democracy', 'Justice' etc. Out of these 136 words, 74 occur in the LibCon data set. Table 3.1 presents the 74 words that

occur in the LibCon data set, with bold-face words representing the words that shifted the most in use between Liberal and Conservative party members. Word shift is computed as in Section 3.5. The complete list of 136 key words is provided in the supplemental section.

Movie Review: The popular movie review (MR) data set from (Pang and Lee (2005)) with 'positive' and 'negative' labels is also used for experiments. However, to keep sizes of all data sets comparable, a random sample of 2500 positive and 2500 negative reviews is used. The full MR data set consists each of 5331 positive and negative samples.

SST: The Stanford Sentiment Tree[3] (SST) bank consists of train, dev and test sets. Data is in the form of reviews with labels indicative of 'very positive', 'positive', 'very negative' and 'negative' sentiment. In our experiments, we binarize labels to just 'positive' and 'negative'. Like with the MR data set, we randomly sample the training data set consisting of 8,545 samples to obtain a smaller training sample of 5000 points. The trained model is then tested on the original test data set consisting of 2210 data points.

Beauty, Book, Music: These data sets consists of 6000 beauty product, book and music reviews with a balanced distribution of three class labels. This data set was introduced by (He et al. (2018a)). Data is obtained by sampling from a larger data set of product reviews obtained from Amazon (McAuley et al. (2015)). Labels are 'positive', 'negative' and 'neutral'. Since the algorithm introduced by the authors is a transfer learning set up, reviews from a source domain of say books, music and electronics is used to predict labels on the target domain like beauty. The same is repeated for different source and target domains. Performance of the proposed algorithm is compared against the average accuracy reported on each domain.

Baselines

For all test data sets, in addition to contrasting the performance of the proposed adaptation layer added to a CNN and BiLSTM encoder, we also present comparisons

[3] https://nlp.stanford.edu/sentiment/

action	cost	dream	help	media	**reform**	**women**
amendment	court	education	honor	need	**republican**	work
attack	**crisis**	fact	hope	order	right	world
budget	deal	**force**	income	pledge	risk	
burden	debate	**freedom**	information	**police**	rule	
business	**debt**	fund	**insurance**	poll	school	
candidate	decision	funding	**justice**	power	spending	
care	defense	future	**labor**	president	**state**	
class	deficit	generation	leader	problem	**truth**	
college	**democrat**	**government**	leadership	program	value	
congress	development	**governor**	**legislature**	protection	**violence**	
control	divide	health	**legislation**	race	wealth	

Table 3.1: This table presents 74 words representing key political concepts common to Liberal and Conservative users on Twitter. Words in bold are the ones that 'shift' the most in use between Liberal and Conservative users on Twitter.

against 'Vanilla' CNN and BiLSTM encoders. For popular data sets like the MR and SST data sets we present results against some additional baselines, all of which are variants of the basic vanilla RNN encoders. The baselines that we used in this paper are as follows

1. **BoW:** This is a standard baseline in which each sentence is expressed as a weighted sum of its constituent word embeddings. Weights used are raw word counts. Both generic (GloVe) and DA word embeddings are used in this baseline.

2. **Vanilla CNN:** A CNN based sentence classification framework as introduced by (Kim (2014b)) is used as a 'Vanilla' CNN baseline.

3. **Vanilla BiLSTM:** A Basic BiLSTM encoder followed by max pooling as proposed by (Conneau et al. (2017b)) is used as the 'Vanilla' BiLSTM baseline. Note that the vanilla baseline contains no additional features like attention.

4. **BERT:** Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. (2018)) is a generic encoder framework that learns sentence embeddings by jointly conditioning on both left and right context in all layers. The

Bert encoder is generic and requires a tuneable output layer to be learned in order to perform a chosen task. In our work, we use the pooled output from the Bert encoder to obtain sentence representations for all our data and then learn weights for a classifier to perform classification on the given data set.

5. **Self-attention:** A model that generates interpretable, structured sentence embeddings using Self-attention mechanism (Lin et al. (2017)).

6. **LR-Bi-LSTM:** A model that imposes linguistic roles to Bi-LSTM architectures (Qian et al. (2017)).

7. **DAS:** The Domain Adaptive Semi-supervised algorithm (He et al. (2018a)) is a transfer learning based solution that performs sentiment classification. In their experiments, the authors train on source domains that include book, music and electronics reviews, and tests on a target domain consisting of reviews of beauty products (Beauty). In contrast, our method does not train on these three sources; rather, it uses only the Beauty reviews to learn the DS word embeddings and then combines these with the generic pretained embedding to learn the DA embedding used for classification on the Beauty domain. The same procedure is repeated for other domains as well. Metric reported on this data set is Accuracy, the same used by (He et al. (2018a)). We compare against the average accuracy obtained on each domain.

Experimental methodology

The LibCon, MR and Beauty, Book and Music data sets do not have dedicated train/dev/test splits, so we created $80/10/10$ splits. The SST dataset has a predefined train/dev/test splits. On the SST data set, the original test data set of 2210 data points is used for testing. Training data is determined by randomly sampling the original training data of 5000 points, in order to mimic a setting in which the training and test data sets are not too large as in (K Sarma et al. (2018)). When pre-tuned models are available, such as for BERT, we further fine-tune it using our train/dev splits. All the baselines use the same dataset splits.

Hyperparameters: All word embeddings-GloVe, DS and KCCA projections used to obtain the DA embeddings are of dimension 300. Both the CNN and BiLSTM encoders learn sentence embeddings of 300 dimensions on all data sets, except on the Beauty data set where the CNN encoders have a filter size of 120. Results reported in Section 3.6 are obtained using the 'non-static' mode of the CNN encoder. Dropout probabilities are determined empirically. When using the CNN with multi-class data sets, we follow the same hyperparameters as (He et al. (2018a)). CNN encoder filters are each of size 100. Dropout probability is 0.5 and max norm is 3. The optimizer used is RMSprop with a learning rate of 0.0005. Networks are trained for 30 epochs with early stopping when using end-to-end CNN architecture.

When using BERT, the 'bert-base-uncased' model is used to obtain sentence embeddings. Pooled output from the BERT encoder, that represents the sentence embeddings has 768 dimensions; this dimension is not reduced in our experiments. The classifier learned during tuning on the train data sets is similar to the linear classification layer introduced in the BERT classification framework, with the exception of an additional Tanh activation. The additional Tanh activation function was used inorder to obtain comparable results for BERT and other baselines. The size of test dates is particularly challenging for a large model like BERT that has several learnable parameters, even when used only for fine-tuning the pre-trained model. The BERT classification framework is reproduced for use in our experiments, reflecting the original tuning scripts to the best of our ability. Additional information about hyperparameters can be found in the supplement.

Table 3.2 presents results on the LibCon, Beauty, Book and Music data sets and Table 3.3 presents results on the SST and MR data sets. The performance metric reported in both tables is accuracy.

Results

From Table 3.2, it is observed that on the LibCon data set, where we have a considerable difference in language use between the two groups of users, the adapted BiLSTM and adapted CNN perform much better than the vanilla baselines. Fur-

Algorithm	Acc on LibCon	Acc on Beauty (B)	Acc on Book (B)	Acc on Music (B)	Beauty (I)		Book (I)		Music (I)	
					Acc	F-score	Acc	F-score	Acc	F-score
BoW (GloVe)	64.0	56.18	69.1	66.5	74.8	74.8	66.08	66.08	75.8	75.8
BoW (DA embeddings, $\alpha = \beta = 0.5$)	65.3	59.02	71.9	67.6	75.12	75.12	67.0	67.0	77.12	77.12
Vanilla CNN	61.36	61.7	66.5	61.5	74,5	74.6	76.8	78.1	77.5	77.5
Vanilla BiLSTM	68.5	75.6	77.6	80.0	84.3	84.3	83.4	83.4	84.2	84.2
Adapted CNN	63.24	62.8	71.0	63.5	80.1	75.0	78.0	76.6	78.1	78.1
Adapted BiLSTM	**72.1**	**77.3**	**80.3**	**81.5**	**85.2**	**85.2**	**84.7**	**84.7**	**85.6**	**85.6**
BERT	70.3	-	-	-	-	-	-	-	-	-
DAS	N/A	56.0	67.6	58.6	54.8	-	61.06	-	55.1	-

Table 3.2: This table presents performance (accuracy score) of the baseline algorithms on the LibCon and Beauty, Book and Music data sets in balanced and imbalanced setting. Micro F-score is reported on the imbalanced Beauty, Book and Music data sets as well. Bold face indicates best performing algorithm.

thermore, our proposed adaptation layer improves the performance of the Vanilla BiLSTM to surpass the performance of fine-tuned BERT. Note that the BERT encoder is pre-trained on all of Wikipedia, a much larger training data set than used for training the Adapted BiLSTM encoder.

The adapted BiLSTM and CNN encoders also outperform DAS on the Beauty, Book and Music data sets, all which have three degrees of polarity i.e positive, negative and neutral. All three data set exist in a balanced (B) as well as an imbalanced (I) setting. In the balanced framework, there is an equal distribution of the three classes in the ata set. In the imabalanced setting, roughly 80% of the reviews come from the positive class, while the remaining 20% is made up of neutral and negative reviews. This shows that the shallow adaptation framework is applicable even in multiclass settings with finer resolutions of sentiment polarity.

From Table 3.3, note that on the MR data set, the performance of fine-tuned BERT is lower than the BiLSTM baselines. While the size of training data used to optimize BERT, makes the encoder an excellent generic sentence encoder, the small size of the tuning data sets poses a considerable challenge. This is because, when the BERT encoder is fine tuned for a particular task, output from the encoder is treated as a fixed feature on top of which it learns a task specific layer. Using pre-trained BERT provides a high quality feature vector that can be used for applications on generic data sets, but this does not assure good performance when tuned on a small data set where words may be used idiosyncratically. On the other hand, on

Algorithm	Accuracy on MR	Accuracy on SST
BoW (Generic)	75.7*	48.9*
BoW (DA embeddings)	77.0*	49.2*
Vanilla CNN	72.5*	49.06*
Vanilla BiLSTM	81.8*	50.3*
LR-Bi-LSTM	82.1	50.6
Self-attention	81.7	48.9
Adapted CNN	80.8*	50.0*
Adapted BiLSTM	**83.1***	51.2
BERT	74.4*	**51.5**

Table 3.3: This table presents results for the MR and SST data sets. Reported performance metric is accuracy. Results with * indicate that performance metric is reported on the test dataset after training on a subset of the original data set.

Algorithm	Accuracy on MR(1000)	Accuracy on SST(1000)
Vanilla CNN	65.1	48.8
Adapted CNN	74.7	49.6
Vanilla BiLSTM	76.9	50.08
Adapted BiLSTM	78.1	50.3

Table 3.4: This table presents the accuracy obtained by Vanilla and adapted baselines on smaller subsamples of the training data for the MR and SST data sets each containing 1000 data points.

a highly polarized data set like the LibCon data set, it is likely that the optimized BERT encoder captures the distinct linguistic features in the sentence embeddings.

Varying Size of Training Data:

To further illustrate the effectiveness of our method on data sets with limited training data, we train the Vanilla and adapted BiLSTM and CNN encoders with even smaller samples of the training data. Tables 3.4 and 4.5 compares accuracy on the MR and SST data sets, of Vanilla and adapted encoders trained with 1000 and 2500 points respectively. Repeating this experiment with BERT resulted in tremendous over-fitting on the test set and so we do not present results from BERT in Tables 3.4 and 4.5.

Results Table 3.4 strengthens our hypothesis that the adaptation layer is particu-

Algorithm	Accuracy on MR(2500)	Accuracy on SST(2500)
Vanilla CNN	66.5	49.0
Adapted CNN	77.4	50.7
Vanilla BiLSTM	78.8	50.2
Adapted BiLSTM	80.1	51.0

Table 3.5: This table presents the accuracy obtained by Vanilla and adapted baselines on smaller subsamples of the training data for the MR and SST data sets each containing 2500 data points.

larly well suited for small training and test data regimes.

3.7 Conclusions and Future Work

Results in this chapter demonstrate how domain semantics captured in adapted word embeddings can be used to improve the performance of (pre-trained) encoders in downstream tasks such as sentiment classification, particularly on data sets where obtaining training data sufficient enough for tuning pre-trained encoders or for end-to-end training is difficult. Experiments show the effectiveness of the method in binary as well as multiclass classification tasks. The proposed framework also outperforms competing transfer learning based algorithms in overcoming limitations posed by the size of training data, while learning fewer parameters than an end-to-end trained network along with obtaining better results on classification tasks. Most recent work that focusses on capturing domain semantics when learning word embeddings, demonstrate the effectiveness of such approaches via performance on a downstream task such as sentiment analysis/classification. There is little analysis performed to determine if the proposed techniques are indeed capturing relevant domain information. In Sections 3.5 and 3.6 we provide a simple yet effective demonstration of how techniques such as KCCA, that are used to capture domain semantics, can capture relevant semantics in the learned word embeddings.

As future work, we will interface the adaptation layer for use recently introduced large scale language models such as BERT. Due to size and memory complexities

of available implementation of large scale language models such as BERT, an easy integration with our proposed adaptation framework does not currently exist. Yet another extension for this work is to model the shallow adaptation via soft attention layers when training a network end-to-end.

4 SIMPLE ALGORITHMS FOR SENTIMENT ANALYSIS ON SENTIMENT RICH, DATA POOR DOMAINS.

4.1 Overview

Unlike the un/weakly supervised embedding algorithms proposed in the previous chapters, this chapter introduces a supervised algorithm to learn 'polarized' word embeddings. This algorithm first appeared as in (Sarma and Sethares (2018)). While the previously proposed algorithms focus on capturing domain semantics, this algorithm works on incorporating document label information in word representations. In addition to results from sentiment classification tasks, word embeddings learned from the proposed algorithm capture word antonym pairs.

4.2 Abstract

Standard word embedding algorithms learn vector representations from large corpora of text documents in an unsupervised fashion. However, the quality of word embeddings learned from these algorithms is affected by the size of training data sets. Thus, applications of these algorithms in domains with only moderate amounts of available data is limited. In this paper we introduce an algorithm that learns word embeddings jointly with a classifier. Our algorithm is called SWESA (Supervised Word Embeddings for Sentiment Analysis). SWESA leverages document label information to learn vector representations of words from a modest corpus of text documents by solving an optimization problem that minimizes a cost function with respect to both word embeddings and the weight vector used for classification. Experiments on several real world data sets show that SWESA has superior performance on domains with limited data, when compared to previously suggested approaches to word embeddings and sentiment analysis tasks.

4.3 Introduction

Word embedding algorithms learn vector representations for words that are useful to quantify semantic relationships between words in a given text. Additionally, word embeddings are used to initialize several algorithms for sentiment analysis, sentence encoding. Currently popular embedding algorithms such as word2vec (Mikolov et al. (2013c); Le and Mikolov (2014)), GloVe (Pennington et al. (2014a)), are based off neural network methods and have achieved tremendous success in various word, sentence/document level evaluation tasks. These algorithms thrive on the large volumes of training data sets when learning high quality word embeddings. However, there is an increase in application domains where getting large amounts of data is not always possible. Furthermore, in some of these domains, representing words from off-the-shelf word embeddings such as ones obtained from training word2vec, GloVe on Wikipedia or common-crawl may not be efficient. This is because the sentiment expressed by a word in such datasets could be somewhat different from the sentiment expressed by the same word when it appears in Wikipedia, common-crawl. A concrete example of such domains is given in the next paragraph. *In such cases off-the-shelf word embeddings are not very useful and better techniques are required to learn word embeddings and classifiers for sentiment analysis.*

The goal of this paper is to build classification algorithms for sentiment analysis on small, domain specific data sets that are sentiment rich. An example of such a data set is the Substance Use Disorder (SUD) data set (Mohr et al. (2013)), (Moore et al. (2011)) obtained from digital health intervention treatments. These treatments aim to predict relapse risk by analyzing the content of participants' text messages. Though forum moderators can monitor and provide support when participants are struggling, considerable labor is involved in reviewing and deciding the risk level of each text message. Text data obtained from these discussion forums is rich in sentiments such as 'determination,' 'pleasure,' 'anger,' 'fear' and by analyzing discussion messages for sentiments such as 'anger', 'fear' it is possible to develop efficient algorithms that can automatically tag if a message is positive (i.e. benign) or negative (indicative of relapse).

One approach to this problem would be to use off-the-shelf word embeddings to learn document embeddings and then run a classification algorithm on the obtained embeddings. While this approach does decently well it fails to capture and exploit the semantics of within domain words. For example, consider words such as 'alcohol,' 'holiday,' 'party.' Such words are typically neutral or positive sentiment in the Wikipedia corpus. However, with data sets such as SUD, these words are indicative of a moderate/strong negative sentiment. As a result using off-the-shelf word embeddings from GloVe, word2vec embeddings trained on Wikipedia corpus for building a sentiment analyzer will result in a sentiment analyzer that performs poorly. An alternative is to obtain word embeddings by training word2vec or GloVe on the target dataset, and then using the resultant embeddings for sentiment analysis. Unsurprisingly this approach succeeds only when one has the luxury of large datasets. However, in the SUD application, as mentioned above, the amount of data that is available is limited and hence this approach is not a successful alternative. In fact we argue that in the presence of limited, but sentiment rich data, it is better to learn word embeddings in a supervised manner. This way the resulting embeddings tend to be polarity-aware and are better suitable for downstream tasks such as sentiment analysis. Our contributions are as follows,

1. We introduce an algorithm (Section 4.4) that jointly learns word embeddings as well as a sentiment analyzer (classifier) by solving a bi-convex optimization problem. Our algorithm is called Supervised Word Embedding for Sentiment Analysis (SWESA) . This is an iterative algorithm that minimizes a cost function for both a classifier and word embeddings under unit norm constraint on the word vectors.

2. SWESA uses document labels for learning word embeddings. Using document labels within SWESA helps us overcomes the problem of small-size training data and allows learning of polarity-aware word embeddings. Via a thorough empirical evaluation (Section (4.5)) we show that our algorithm outperforms classifiers built by re-training off-the-shelf word embeddings such as word2vec, GloVe. We also compare SWESA against an algorithm that

uses convolutional neural networks to represent sentences (Kim (2014a)) for sentiment analysis, and a sentiment analysis algorithm based on recurrent neural networks (Socher et al. (2013)). On the A-CHESS dataset where data is limited but rich in sentiment, SWESA outperforms all algorithms by at least 12% on the precision.

3. To demonstrate the fact that the word embeddings learned by SWESA are better than embeddings learned from unsupervised learning algorithms we investigate the polarity of various word embeddings. We show that the embeddings learned from SWESA perform well on antonym task. For example, 'Awful/Good' is the antonym pair returned via SWESA as opposed to 'Awful/Could' obtained via word2vec. SWESA learns such antonym pairs, using document polarities, and as a byproduct of our optimization based formulations, independent of an antonym pairs training data set.

Notation: Throughout this paper we shall denote word vectors as $\mathbf{w}_j \in \mathbb{R}^k$, for $j = 1, \ldots, V$, where V indicates the size of the vocabulary. The matrix of word vectors is $\mathbf{W}$ where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_V] \in \mathbb{R}^{k \times V}$. The classifier to be learned is represented by $\boldsymbol{\theta} \in \mathbb{R}^k$, weights of word vectors $\mathbf{w}_j$ in document i are contained in the vector $\boldsymbol{\phi}_i \in \mathbb{R}^V$, and the document label of the i^{th} document is indicated by y_i, document i is represented as $d_i = \mathbf{W}\,\boldsymbol{\phi}_i$. Let $\Phi = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_N] \in \mathbb{R}^{V \times N}$ be the matrix containing weight vectors $\boldsymbol{\phi}_i$ and vector $\mathbf{y} = [y_1, y_2, ...y_N]$ be the vector containing document labels.

4.4 Supervised Word Vectors for Sentiment Analysis

Given a collection of documents $d_1, d_2, \ldots d_N$ with binary sentiments $y_1, y_2, \ldots, y_N$ respectively, the aim is to learn a classifier that when given a new, previously unseen document d can accurately estimate the sentiment of the document. There could be class imbalance in the training data and so the algorithm should explicitly account for such a class imbalance. We approach this problem by introducing a new algorithm called SWESA. SWESA simultaneously learns word vector embeddings and

a classifier, by making use of document polarity/sentiment labels. Representation of documents within SWESA is motivated by the fact that in short texts like "I am sad", "I am happy", polarity of the sentence hinges on the words "sad" and "happy". As a result, by learning polarity aware word embeddings, good vector representations for documents can be achieved. For instance, in the above example, the distance between the vectors $(\mathbf{w}_I + \mathbf{w}_{am} + \mathbf{w}_{sad})$ and $(\mathbf{w}_I + \mathbf{w}_{am} + \mathbf{w}_{happy})$ would capture dissimilarities in sentiment of these two documents while at the same time reflecting similarities in sentence structure.

Text documents in this framework are represented as a weighted linear combination of words in a given vocabulary. Weights can be either the term frequencies (tf) of words within each document or term frequency-inverse document frequency (tf-idf). Weights provided as input to SWESA for experiments described in Section (5.6) are term frequencies. We choose this weighting scheme to mimic the concept of local context used in the word2vec family of algorithms. Global co-occurrence information can be leveraged by using tf-idf for weighting words in documents. Such an approach in not entirely unheard off in sentiment analysis tasks, where word embeddings are considered as features for a classification algorithm (Labutov and Lipson (2013)).

SWESA aims to find vector representations for words, which when combined using weights, in a bag-of-words framework provide us with embeddings for text documents. These embeddings should be such that applying a nonlinear transformation f to the product $(\theta^\top \mathbf{W}\, \boldsymbol{\phi})$ results in a binary label y indicating the polarity of document. Mathematically we assume that,

$$\mathbb{P}[Y = 1 | d = \mathbf{W}\, \boldsymbol{\phi}, \boldsymbol{\theta}] = f(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}) \tag{4.1}$$

for some function f. In order to solve for $\boldsymbol{\theta}$ and $\mathbf{W}$, a regularized negative likelihood minimization problem is solved. This optimization problem is as (4.1) and can be

solved as a minimization problem with objective function,

$$J(\boldsymbol{\theta}, \mathbf{W}) = \frac{-1}{N}\Big[\mathrm{C}_{+} \sum_{y_i=+1} \log \mathbb{P}(Y = y_i | \, \mathbf{W}\, \boldsymbol{\phi}_i, \boldsymbol{\theta}) + \mathrm{C}_{-} \sum_{y_i=-1} \log \mathbb{P}(Y = y_i | \, \mathbf{W}\, \boldsymbol{\phi}_i, \boldsymbol{\theta})\Big] + \lambda_{\boldsymbol{\theta}} || \, \boldsymbol{\theta} \, ||_2^2. \quad (4.2)$$

This optimization problem can now be written as

$$\min_{\substack{\boldsymbol{\theta} \in \mathbb{R}^k, \\ \mathbf{W} \in \mathbb{R}^{k \times V}}} \quad J(\boldsymbol{\theta}, \mathbf{W}) \quad (4.3)$$
$$\text{s.t.} \; || \, \mathbf{w}_j \, ||_2 = 1 \; \forall j = 1, \ldots V.$$

The vector $\boldsymbol{\phi}_i$ is a vector of weights, corresponding to the different words, for document d_i. As mentioned previously, for testing SWESA term frequencies of different words in a certain document i are used in $\boldsymbol{\phi}_i$. $\lambda_\theta > 0$ is the regularization parameter for the classifier θ, C_+ is the cost associated with misclassifying a document from the positive class and C_- is the cost associated with misclassifying a document from the negative class. Following the heuristic suggested by (Lin et al. (2002)), $\mathrm{C}_+ = \frac{\mathrm{N}_-}{N}$ and $\mathrm{C}_- = \frac{\mathrm{N}_+}{N}$, where N_+ is the number of positive documents in the corpus and N_- is the number of negative documents in the corpus. This scheme is particularly useful when dealing with data sets with imbalanced classes. When using a balanced data set $\mathrm{C}_+ = \mathrm{C}_-$. Sentiment in a given document is captured by the document label y_i, which in this framework is a binary label that capture sentiments such as 'positive/negative' or 'threatening/benign' depending on the data set.

The unit norm constraint in the optimization problem shown in (4.3) is enforced on word embeddings to discourage degenerate solutions of $\mathbf{w}_j$. For example in the absence of this constraint, the optimal $\mathbf{w}_j^*$ is typically a vector of zeros. Note that this optimization problem is bi-convex, but it is not jointly convex in the optimization variables. Algorithm 1 shows the algorithm that we use to solve the optimization problem in (4.3). This algorithm is an alternating minimization proce-

dure that initializes the word embedding matrix $\mathbf{W}$ with $\mathbf{W}_0$ and then alternates between minimizing the objective function w.r.t. the weight vector $\boldsymbol{\theta}$ and the word embeddings $\mathbf{W}$.

Algorithm 1 Supervised Word Embeddings for Sentiment Analysis (SWESA)

Require: $\mathbf{W}_0$, Φ, C_+, C_-, λ_θ, $0 < k < V$, Labels: $\mathbf{y} = [y_1, \ldots, y_N]$, Iterations: $T > 0$,

1: Initialize $\mathbf{W} = \mathbf{W}_0$.
2: **for** $t = 1, \ldots, T$ **do**
3: Solve $\boldsymbol{\theta}_t \leftarrow \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \mathbf{W}_{t-1})$.
4: Solve $\mathbf{W}_t \leftarrow \arg\min_{\mathbf{W}} J(\boldsymbol{\theta}_t, \mathbf{W})$.
5: **end for**
6: Return $\boldsymbol{\theta}_T, \mathbf{W}_T$

Logistic regression model

The optimization problem in (4.2) assumes a certain probability model and minimizes the negative log-likelihood under norm constraints. While, the specific goal of the user might dictate an appropriate choice of probabilistic model, for a large class of classification tasks such as sentiment analysis, the logistic regression model is widely used. In this section we assume that the probability model of interest is the logistic model. Under this assumption the minimization problem in Step 3 of Algorithm 1 is a standard logistic regression problem [1]. Many specialized solvers have been devised for this problem and in this implementation of SWESA, a standard off-the-shelf solver available in the scikit-learn package in Python is used. In order to solve the optimization problem in line 4 of Algorithm 1 a projected stochastic gradient descent (SGD) with suffix averaging (Rakhlin et al. (2011)) is used. In suffix averaging the last few iterates obtained during stochastic gradient descent are averaged. Suffix averaging guarantees that the noise in the iterates is reduced and has been shown to achieve almost optimal rates of convergence for minimization of strongly convex functions. For experiments in Section 5.6 we set $\tau = 50$.

[1] A bias term, γ can be trivially introduced in the logistic regression model.

Gradient updates for $\mathbf{W}$ given $\boldsymbol{\theta}$ are of the form,

$$\nabla J(\boldsymbol{\theta}, \mathbf{W}) = \frac{1}{N}\Big[\sum_{y_i=+1} \frac{-\,\mathrm{C}_+\, y_i\, \boldsymbol{\theta}\, \boldsymbol{\phi}_i^\top}{1 + e^{y_i(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}_i)}} + \sum_{y_i=-1} \frac{-\,\mathrm{C}_-\, y_i\, \boldsymbol{\theta}\, \boldsymbol{\phi}_i^\top}{1 + e^{y_i(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}_i)}}\Big]. \tag{4.4}$$

Algorithm 2 implements the SGD algorithm (with stochastic gradients instead of full gradients) for solving the optimization problem in step 4 of Algorithm 1.

Algorithm 2 Stochastic Gradient Descent for $\mathbf{W}$

Require: $\boldsymbol{\theta}, \gamma, \mathbf{W}_0$, Labels: $\mathbf{y} = [y_1, \ldots, y_N]$, Iterations: N, step size: $\eta > 0$, and suffix parameter: $0 < \tau \leq N$.

1: Randomly shuffle the dataset.
2: **for** $t = 1, \ldots, N$ **do**
3: Set $C_t = C_+$ if $y_t = +1$, $C_t = C_-$ if $y_t = -1$.
4: $\widetilde{\mathbf{W}}_{t+1} = \mathbf{W}_t - \frac{\eta C_t}{1+e^{y_i(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}_i)}} \times (-y_i\, \boldsymbol{\theta}\, \boldsymbol{\phi}_i^\top)$
5: $\mathbf{W}_{t+1,j} = \mathbf{W}_{t+1,j} \,/\!/\, \mathbf{W}_{t+1,j} \,||_2 \; \forall j = 1, 2, \ldots, V$
6: $\eta \leftarrow \frac{\eta}{t}$
7: **end for**
8: Return $\mathbf{W} = \frac{1}{\tau} \sum_{t=N-\tau}^{N} \mathbf{W}_t$

Sketch of Convergence Analysis of SWESA:

At a high level, SWESA can be seen as a variation of the supervised dictionary learning problem (SDL). Within SDL (Mairal et al. (2009)) given labeled data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, and unlabeled part of the data that lies in a d dimensional space, the goal is to learn a dictionary D of size $d \times k$, $(k >> d)$ such that each $x_i = Dz_i$ where z_i is a sparse encoding of x_i w.r.t. dictionary D. Further, the label is generated by a linear classifier w.r.t z_i, i.e. $y_i = \mathbf{W}^\top z_i$. The learning problem is to estimate the dictionary, the codes of each data point and the classifier. SWESA can be roughly mapped to the SDL by considering dictionary D of size $k \times V$, where each column corresponds to a word embedding. However, there are significant differences between SWESA and SDL.

Significant differences between SDL and SWESA are i) Input to SDL is a labeled dataset, with each data point already represented as a vector. This allows for a definition of reconstruction error within algorithms designed for SDL. In contrast, SWESA has labeled unstructured data without a direct vector representation, and the aim is to learn vector representations for such data. As a result the notion of reconstruction error used in SDL does not apply to SWESA and hence the optimization formulation used is significantly different from the one used in SDL. ii) In SDL sparse encoding of each data point is to be learned, whereas in SWESA this sparse encoding is known and is proportional to the number of times a word appears in the document. (iii) Finally, in SDL the classifier is a high-dimensional vector that acts on the latent codes.

Despite these differences, with suitable modifications convergence analysis for SWESA can be performed. Extensive literature on alternative minimization and their convergence guarantees for SDL and related problems such as matrix completion (Jain et al. (2013)) exist. A full theoretical analysis is out of scope of this paper and is left for future work.

Initialization of W

Two different initialization procedures are used to obtain $\mathbf{W}_0$. The first method uses the Latent Semantic Analysis (Dumais (2004)) (LSA) procedure to form the matrix of word vectors $\mathbf{W}_0$ from the given corpus of text documents. The second method uses the word2vec (w2v) algorithm to form word vector matrix $\mathbf{W}_0$ from the corpus.

4.5 Experimental Evaluation and Results

We now describe in brief the data sets and the algorithms that we use to evaluate SWESA.

Data Sets

We conduct our experiments on four small sized data sets out of which 3 are balanced and 1 is imbalanced. All four data sets are drawn from different domains and hence have different vocabularies and contexts. The four data sets are:

- **Yelp:** This data set consists of 1000 restaurant reviews labeled 'positive' or 'negative'. There are a total of 2049 distinct word tokens in this data set.
- **IMDB:** This data set consists of 1000 reviews of movies with labels 'positive' or 'negative'. There are a total of 3075 distinct word tokens.
- **Amazon:** This data set contains 1000 product reviews with labels 'positive' or 'negative'. This data set has 1865 distinct word tokens.
- **A-CHESS:** This is a proprietary data set[2] and is obtained from an intervention treatment aimed at alcohol disorder. Text is obtained via the discussion forums part of the A-CHESS mobile app. There are a total of 2500 messages with 8% of the messages indicating relapse risk or 'threat'. Typical messages from this data set are of the form, *"I've been clean for 7 months now, but I still feel I may not make it"*. Such a message is marked as 'threat'. Positive or 'benign' messages are of the form, *"30 days and sober! I feel great!!"*. This labeling is human moderated. The goal is to automate this process. After removing special characters, this data set consists of 3400 distinct word tokens.

The first three data sets are obtained from Kotzias et al. (2015).

Baselines

We choose two types of baselines that cover both neural and non-neural network style algorithms.

Standard baselines:

[2]Center for Health Enhancement System Services at UW-Madison.

1. **Two-Step (TS):** The TS algorithm is a family of algorithms where we perform two steps (i) First, word embeddings are trained in an unsupervised manner on the target dataset. These word embeddings are combined together with appropriate weights (as is done in SWESA) to obtain document embeddings. (ii) In the second step the document embeddings obtained from the first step are used in a logistic regression classifier to obtain a classification model. To implement our first step we use two types of word embeddings, namely LSA word embeddings, and word2vec (re-w2v) obtained by re-training LSA, and word2vec algorithms on our target datasets. We use an open source implementation in gensim[3] to get word2vec embeddings.

2. **Naive Bayes classifier:** The classic Naive Bayes classifier for sentiment classification based on the Bag-of-words features, optimized in NLTK toolkit in Python is used.

Neural network based baselines:

1. **Recursive Neural Tensor Network (RNTN):** RNTN is an RNN proposed by Socher et al. (2013) that learns compositionality from text of varying length and performs classification in a supervised fashion with fine grained sentiment labels. Since SWESA is aimed at binary classification, RNTN is also used in a binary classification framework. RNTN has been shown to perform better than the previously proposed Recursive Auto Encoder (RAE) by Socher et al. (2011) and hence we limit our comparisons to RNTN only.

2. **CNN based sentence classification:** Kim (2014a) propose a Convolutional Neural Network (CNN) based architecture that takes as input word embeddings and uses a CNN based architecture to learn sentence embeddings. These sentence embeddings are then used for classification. This algorithm belongs to a large class of algorithms that take as input word embedding and learn sentence embeddings for other tasks. While it seems that SWESA is similar to these algorithms, the significant difference between the two lies

[3] `https://radimrehurek.com/gensim/models/word2vec.html`

in the application domain of SWESA. SWESA is geared towards small sized datasets, whereas neural network based algorithms are data intensive and can extract useful relationships from very large amounts of data.

Note that neural network based baselines are used in two modes,

- **Pre-trained (Pr-tr):** In this framework both RNTN and CNN-static (CNN-S), CNN-non static (CNN-NS) are initialized with word2vec embeddings (w2v) that were trained on the Pang and Lee data set (Pang and Lee (2005)). This data set consists of roughly 10k text documents and is roughly 10 times the size of the data sets considered in experiments in Section 4.6.

- **Re-trained (Re-tr):** In this framework, RNTN is retrained on the data sets described in subsection 4.5. Further, CNN-static and non static are used in the following ways,

    1. CNN-static and CNN-non static are initialized with re-trained w2v (re-w2v) embeddings. The training set is still the Pang and Lee data set used by the authors Kim (2014a). We expect the performance of this baseline to be poorer when compared to CNN-static and non static initialized with pre-trained word embeddings. This baseline is used to investigate the sensitivity of CNNs to initial input word embeddings.
    2. CNN-static and CNN-non static are initialized with pre-trained word2vec embeddings but trained on train/dev sets obtained from the data sets described in subsection 4.5. We expect this data set to be the least well performing data set due to the size of the training data. This baseline is used to illustrate the limitations of small sized training data sets on neural network algorithms.

Dimensionality of word embeddings and hyperparameters

Dimensions of word embeddings and other hyperparameters such as regularization on the logistic regression classifier are determined via 10 fold cross validation. Pre-

trained RNTN[4] and CNN-static/non static[5] are used with the training sets and parameters as described by the authors of both works. Procedure for retraining RNTN as well as for fine tuning word embeddings for use in CNN-non static follow the methods described in Socher et al. (2013) and Kim (2014a) respectively.

4.6 Results from classification tasks

Performance metrics reported are average Precision and average AUC. AUC is only reported for model that provide prediction probabilities in addition to predicted label. For the A-CHESS data set the minor class, i.e 'threat' owing to the large imbalance is of more importance. This is because, the system can accept few false positives, but the risk of misclassifying a 'threat' message as 'benign' has a larger impact. Owing to this, on this data set, the best performing classifier will be one that achieves maximum precision. Note that the objective of these results is to show how well SWESA performs in applications with small sized data sets as compared to i) completely re-training a neural network with a smaller training set, ii) or initializing a neural network with word embeddings learned on small data sets and iii) against a neural network that uses pre-trained word embeddings and larger training sets.

SWESA against pre-trained neural nets initialized with pre-trained word embeddings: On the balanced Yelp, Amazon and IMDB data sets, pre-trained RNTN, CNN-static and non static perform the best. From tables 4.1 and 4.2, observe that the best performing baseline CNN-NS achieves an average precision of 87.98, 87.15 and 87.77 respectively on the three balanced data sets. This result is *not surprising*, given that the pre-trained RNTN and CNNS are i)initialized with pre-trained word embeddings and ii)trained on a data set with roughly 10 times more training data points than within SWESA. However, note that on the imbalanced A-CHESS data set RNTN fails to perform any classification. This can be attributed to the fact that RNTN is a dependency parser and on varying length text like within SWESA, it is very hard to capture dependencies. CNN-static and CNN-non static achieve poor

[4] `https://nlp.stanford.edu/sentiment/code.html`
[5] `https://github.com/yoonkim/CNN_sentence`

precision owing to the class imbalance which is not accounted for when training them.

SWESA against re-trained neural nets: CNN-static (re-w2v) and CNN-non static (re-w2v) perform comparably well to SWESA on the balanced data sets. From tables 4.1 and 4.2 notice that on the balanced Yelp, Amazon and IMDB data sets, SWESA achieves an average precision of 78.35, 80.36 and 77.27 respectively while CNN-non static (re-w2v) achieves an average precision of 78.10, 80.40 and 75.85 respectively. These results suggest that SWESA is a suitable alternative to neural network architectures that are sensitive to initializing word embeddings. This result is particularly useful for data sets that are not large enough to enable learning of high quality word embeddings. Note that on the A-CHESS data set CNN-static and non static perform poorly.

SWESA against neural nets initialized with re-trained word embeddings: re-trained RNTN performs very poorly on the three balanced data sets and fails to classify the A-CHESS test set. On the other hand while CNN-static (re-w2v) and CNN-non static (re-w2v) perform comparably well to SWESA on the balanced data sets, both algorithms perform poorly on the A-CHESS data set. CNN-static and non static achiever average precision of 15.62 and 18.98 respectively on the A-CHESS data set while SWESA achieves an average precision of 35.80. This is not a surprising result because the training data to RNTN and CNNs is much smaller than the Pang and Lee Data set.

4.7 Polarity of word embeddings

The objective of SWESA is to perform effective sentiment analysis by learning embeddings from text documents with sentiment labels. Since, word embeddings are learned via a joint optimization framework one can expect that the resulting word embeddings are polarity aware. To test our hypothesis, we investigate if one can predict the antonym of a word. That is, given a word v whose word embedding is $\mathbf{w}_v$ we determine the antonym to be that word a with word embedding $\mathbf{w}_a$ such that the cosine similarity between $\mathbf{w}_a, \mathbf{w}_v$ is minimized over all possible choices of

Data Set	Method	Avg Precision	Avg AUC
Yelp	SWESA (LSA)	78.09±2.8	86.06±2.4
	SWESA (re-w2v)	**78.35±4.6**	**86.93±3.5**
	TS (LSA)	76.27±3.0	83.05±5.0
	TS (re-w2v)	65.22±4.4	69.08±3.5
	Naive Bayes	57.07±3.3	61.16±4.5
	RNTN (Pre-tr)	83.31±1.1	-
	RNTN (retrained)	51.15±4.3	-
	CNN-S (Pre-tr)	86.45±0.8	-
	CNN-NS (Pre-tr)	87.98±0.5	-
	CNN-S (re-w2v)	76.89±2.5	-
	CNN-NS (re-w2v)	78.10±1.5	-
	CNN-S (Re-tr)	68.55±1.2	-
	CNN-NS (Re-tr)	72.80±0.5	-
Amazon	SWESA (LSA)	80.31±3.3	87.54±4.2
	SWESA (re-w2v)	**80.36±2.8**	**87.19±3.3**
	TS (LSA)	77.32±4.6	85.00±6.2
	TS (re-w2v)	71.09±6.2	77.09±5.3
	Naive Bayes	72.54±6.4	61.16±4.5
	RNTN (Pre-tr)	82.84±0.6	-
	RNTN (retrained)	49.15±2.1	-
	CNN-S (Pre-tr)	87.08±1.0	-
	CNN-NS (Pre-tr)	87.15±0.8	-
	CNN-S (re-w2v)	78.85±1.2	-
	CNN-NS (re-w2v)	80.40±1.0	-
	CNN-S (Re-tr)	70.15±1.8	-
	CNN-NS (Re-tr)	72.86±2.5	-

Table 4.1: This table shows results from a standard sentiment classification task on the Yelp and Amazon data sets. Results from SWESA are in boldface and results from pre-trained models are in blue. No AUC values are reported for models where classification probabilities of classifiers is not available.

Data Set	Method	Avg Precision	Avg AUC
IMDB	SWESA (LSA)	76.40±5.2	81.08±7.6
	SWESA (re-w2v)	**77.27±5.4**	**81.04±6.8**
	TS (LSA)	70.36±5.5	77.54±6.8
	TS (re-w2v)	56.87±7.6	59.34±8.9
	Naive Bayes	73.31±5.6	48.40±2.9
	RNTN (Pre-tr)	80.88±0.7	-
	RNTN (retrained)	53.95±1.9	-
	CNN-S (Pre-tr)	85.54±0.6	-
	CNN-NS (Pre-tr)	87.77±0.5	-
	CNN-S (re-w2v)	72.46±2.0	-
	CNN-NS (re-w2v)	75.85±2.1	-
	CNN-S (Re-tr)	62.07±1.5	-
	CNN-NS (Re-tr)	70.16±1.2	-
A-CHESS	**SWESA (LSA)**	**35.80±2.5**	**83.80±3.1**
	SWESA (re-w2v)	35.40±2.0	83.40±2.6
	TS (LSA)	32.20±3.2	**83.80±3.1**
	TS (re-w2v)	23.60±2.4	68.00±1.2
	Naive Bayes	30.30±3.8	45.23±3.3
	RNTN (Pre-tr)	-	-
	RNTN (retrained)	-	-
	CNN-S (Pre-tr)	18.75±3.2	-
	CNN-NS (Pre-tr)	23.02±2.8	-
	CNN-S (re-w2v)	20.12±2.4	-
	CNN-NS (re-w2v)	25.45±1.8	-
	CNN-S (Re-tr)	15.62±2.7	-
	CNN-NS (Re-tr)	18.98±2.2	-

Table 4.2: This table shows results from a standard sentiment classification task on the IMDB and A-CHESS data sets. Results from SWESA are in boldface and results from pre-trained models are in blue. No AUC values are reported for models where classification probabilities of classifiers is not available.

a. Figure 4.1 shows a small sample of word embeddings learned on the Amazon data set by SWESA and word2vec. The cosine similarity (angle) between the most dissimilar words is calculated and owing to the assumptions on word embeddings, words are depicted as points on the unit circle. From figure 4.1 it is evident that a supervised algorithm like SWESA projects document level polarity onto word level embeddings while an unsupervised algorithm like word2vec that learns embeddings of words via virtue of word co-occurrences will fail to embed polarity. It is important to notice that SWESA learns word polarities by using document

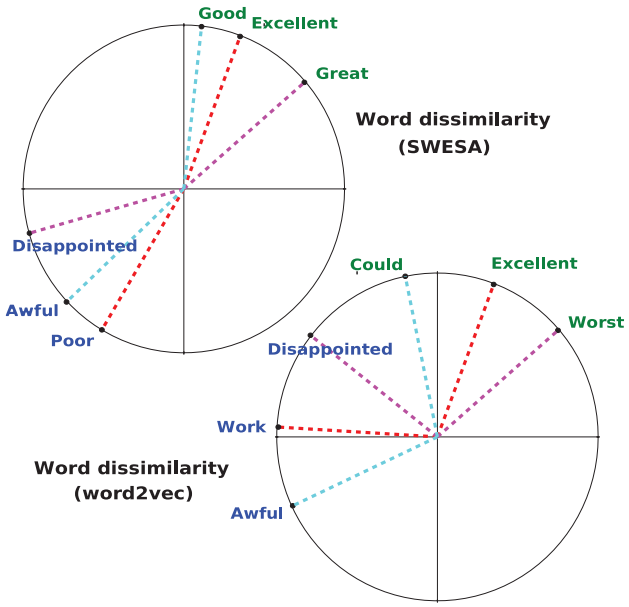

Figure 4.1: This figure depicts word embeddings on a unit circle. Most dissimilar word pairs are plotted based on the cosine angle between the respective word embeddings learned via SWESA and word2vec.

polarities, and these word polarities are useful for antonym tasks. Unlike classical antonym tasks where examples of known antonym pairs are provided, in our setup no such pairs are provided, and yet SWESA was able to do a good job discovering antonym pairs. For example the most dissimilar word to given word 'Excellent' is 'Poor' when learned via SWESA as opposed to 'Work' when learned via word2vec. Thus, word antonym pairs (w_a, w) can be obtained by calculating cosine similarities. Since the aim of experiments in section 5.6 was to perform document level sentiment analysis and observations on word dissimilarities is a consequence of this task, extensive experiments validating word antonym pairs have not been performed. This is left for future work and only a qualitative analysis of learned word embeddings is discussed here.

4.8 Discussions and Conclusions

In this paper we provide a simple optimization based framework, called SWESA that jointly learns word embeddings and a classifier for the task of sentiment analysis. Through extensive experimentation we show that SWESA outperforms both non-neural network algorithms (naive Bayes, LSA) as well as state-of-the-art neural network algorithms based on CNNs and RNNs. As a byproduct of our optimization formulation we show that the word embeddings learned by SWESA are polarity aware and perform very well on antonym tasks, even without being explicitly trained for such tasks. Our contributions strongly emphasize the point that on domains where data is limited but sentiment rich, data size is not a handicap and simple algorithms can do a better task than more involved neural network based algorithm. In the future, we shall investigate modifications of our framework for tasks other than sentiment analysis.

5 MULTI-MODAL SENTIMENT ANALYSIS USING DEEP CANONICAL CORRELATION ANALYSIS

5.1 Overview

This chapter departs from the previous chapters by working with audio and video data in addition to text. For the task of sentiment analysis, we learn multi-modal embeddings that combine correlated information from audio, video and text information using CCA techniques. A combination of each view/modality of data is likely to encode better information than a single view of data. Preliminary results outlined in this chapter reinforce this observation.

5.2 Abstract

This paper learns multi-modal embeddings from text, audio, and video views/-modes of data in order to improve upon downstream sentiment classification. The experimental framework also allows investigation of the relative contributions of the individual views in the final multi-modal embedding. Individual features derived from the three views are combined into a multi-modal embedding using Deep Canonical Correlation Analysis (DCCA) in two ways i) One-Step DCCA and ii) Two-Step DCCA. This paper learns text embeddings using BERT, the current state-of-the-art in text encoders. We posit that this highly optimized algorithm dominates over the contribution of other views, though each view does contribute to the final result. Classification tasks are carried out on two benchmark data sets and on a new Debate Emotion data set, and together these demonstrate that the one-Step DCCA outperforms the current state-of-the-art in learning multi-modal embeddings.

5.3 Introduction

Various social media platforms make available a variety of multi-modal content generated through expression of opinions and ideologies by social media users in the form of written commentary, podcasts, and lifestyle vlogs on a variety of topics such as politics, entertainment, reviews of movies, products etc. Multi-modal data enables one to understand the interplay of linguistic and behavioral cues, particularly when trying to resolve user sentiment or when studying affective behavior, such as the rise of political populism.

Which is more important in human discourse: text, speech, or video?" We approach this question via an experimental paradigm that solves sentiment classification problems using each feature set (text, audio, video) individually, in pairs, and all three together. This allows us to assess the relative importance of the contribution of each data view/mode[1]. By necessity, we investigate ways of "merging" the feature vectors from the three views. The principle result is that more views give better classification, though there is an asymmetry in the development and quality of algorithms for extracting the three views that likely biases any quantitative interpretation of these results.

Recent work on multi-modal (Poria et al. (2018)), (Hu and Flaxman (2018)) and multi-view (Arora and Livescu (2013)) sentiment analysis combine text, speech and video/image as distinct data views from a single data set. The idea is to make use of written language along with voice modulation and facial features either by encoding for each view individually and then combining all three views as a single feature (Poria et al. (2018)), (Hu and Flaxman (2018)) or by learning correlations between views and then combining them in a correlated space (Arora and Livescu (2013)). Each technique has demonstrated significant improvements in classification accuracy when used to detect sentiment. In addition to improving upon performance metrics for a downstream task such as classification, multi-modal data also enables one to study which view (or combination of views) is most efficient in understanding user behavior. For example, when studying populism (Bucy et al.

[1]henceforth we shall use the words view and mode interchangeably

(2018)), visual and tonal expressions of rage have been found to be key characteristics of populist behavior. Since there is a rising interest in using multimodal data for tasks other than sentiment analysis (Hu and Flaxman (2018)), it is important to explore how and to what extent each individual view contributes towards the overall success on a downstream task for a particular data set.

This chapter makes the following contributions: i) Learn multi-modal data embeddings using Deep Canonical Correlation Analysis in a One-Step and Two-Step framework to combine text, audio and video views for the improvement of sentiment/emotion detection. The Two-Step DCCA framework further helps to explore the interplay between audio, video and text features when learning multi-modal embeddings for classification tasks. ii) Encode text using BERT (Devlin et al. (2018)), the current state-of-the-art in text encoders to obtain fixed-length representations for text. There is little literature that uses pre-trained BERT encoders as features without additional fine-tuning. This work adds to the growing body of work that applies BERT as a pre-fixed feature and iii) perform empirical evaluations on benchmark data sets such as CMU-MOSI (Zadeh et al. (2016)) and CMU-MOSEI (Zadeh et al. (2018b))along with a new Debate Emotion data set introduced by (Bucy et al. (2018)).

The rest of this chapter is organized as follows, Section 5.4 presents related work and Section 5.5 describes the methodology used in this chapter. Section 5.6 presents results and Section 5.7 concludes.

5.4 Related Work

The idea of combining multi-modal text, audio and video features expressed in this chapter is closest to that of (Poria et al. (2018)) which encodes text, speech, and visual signals using using a BiLSTM encoder, openSMILE, and 3D-CNN respectively. Encoded outputs are then concatenated and passed through a classifier. In contrast, this chapter employs multi-modal embeddings that are obtained by learning correlated representations of text, audio, and video views using Deep Canonical Correlation Analysis (DCCA) (Andrew et al. (2013)) as in (Dumpala

et al.). This approach is similar to the use of Generalized Canonical Correlation Analysis (GCCA) as in (Rastogi et al. (2015)). Recent work in text based sentiment analysis (K Sarma et al. (2018)–Benton et al. (2016)) has demonstrated the effectiveness of statistical methods like Canonical Correlation Analysis (CCA) and its variants such as GCCA and DCCA on various uni-, bi-, and multi-modal learning tasks.

5.5 Methods

This section briefly reviews Deep Canonical Correlation Analysis (DCCA) and outlines the methods used to obtain unimodal features. This section also outlines the procedure used to obtain the multi-modal embeddings used for experiments in Section 5.6.

Deep Canonical Correlation Analysis (DCCA)

Classic Canonical Correlation Analysis (CCA) (Hotelling (1992)) is a statistical technique used to find a linear subspace in which two sets of random variables with finite second moments are maximally correlated. This idea is applied in the context of multi-modal learning by considering each view/modality of the data to be a random variable, and then using CCA to find the subspace such that non-discriminative features in each view are largely un-correlated, and hence can be filtered out. The natural generalization of this idea, to learning the subspace via non-linear projections obtained from feed forward neural networks, is called Deep CCA.

A DCCA network has input (x_1, x_2), which denotes two input views (corresponding to the same input). Let

$$f_i(x_i; \theta_i) = s_i(W_d h_{d-1} + b_d)$$

denote the final layer of a d layered neural network, whose first layer is $h_i = s_i(W_i x_i +$

b_i) with input x_i. Thus $f_1(x_1; \theta_1)$ and $f_2(x_2; \theta_2)$ represent two neural networks used to encode the two views (x_1, x_2) of the data and are parameterized by $\theta_1 = (W_{1,d}, b_{1,d})$ and $\theta_2 = (W_{2,d}, b_{2,d})$. The objective of DCCA is to determine the parameters of the two networks such that

$$(\theta_1^*, \theta_2^*) = \text{argmax}_{\theta_1, \theta_2} \text{corr}(f_1(x_1; \theta_1), f_2(x_2; \theta_2)) \tag{5.1}$$

where corr denotes the statistical correlation between x_1 and x_2.

Unimodal Feature Extraction

- **Text Encoding:** To encode text in the data, we use pre-trained BERT (Bidirectional Encoder Representations from Transformers). Like the name suggests, BERT is a transformer based language model that conditions jointly on the left and right of a given word. Typically, the BERT encoder is fine-tuned to a particular task by learning an additional task-specific weight layer. We use the output from the BERT encoder, pre-trained on a large corpus of Wikipedia+Book corpus data, and do not perform additional fine-tuning. The choice of encoder is motivated by the success of BERT in achieving the state-of-the-art in several NLP tasks such as sentiment analysis, question-answering, textual entailment etc. Input text in Section 5.6 is encoded using BERT-base, and all text embeddings are of size 768. Henceforth, denote embedded text by $\mathbf{v}_t$.

- **Audio Encoding:** Audio features from data are extracted using COVAREP (Degottex et al. (2014)), that extracts MFCC, pitch, peak slope, and other acoustic features from audio frames. Audio embedding for each video is the average of the audio vectors extracted from each individual frame feature. Resultant audio embeddings are of size 74. Henceforth, audio embeddings are denoted by $\mathbf{v}_a$.

- **Video Encoding:** Framewise features from video stream are extracted using

a combination of FACET[2] and OpenFace 2.0 (Baltrušaitis et al. (2016))[3]. For each 10 second duration video, video-level feature vectors are obtained by averaging across the feature vectors corresponding to individual frames. Video embeddings are denoted by $\mathbf{v}_v$. For two (MOSI and MOSEI) of the three data sets considered in Section 5.6 video features are extracted by using FACET. On the Debate Emotion data set, video features of 2016 debate videos are represented as facial action units features as in (Baltrušaitis et al. (2015)). Resultant video embeddings are of size 35. Henceforth, video embeddings are denoted by $\mathbf{v}_v$.

Methodology

DCCA accepts two views of data at a time and learns a correlated subspace. Since we are working with three views of data, all three views must be combined. We consider two different procedures. The One-Step DCCA concatenates the audio and video features and applies DCCA to this combined audio-video feature and the text features. The Two-Step DCCA combines two of the views in the first step, and then combines the third with the first two in its second step. These are briefly explained in the following sections.

One-Step DCCA

In this set up, one input view to DCCA is fixed to be text encoded ($\mathbf{v}_t$) by BERT and the other input view to the DCCA algorithm is a concatenation of the audio and video embeddings ($\mathbf{v}_a \mid \mathbf{v}_v$). The intuition behind this is that, most often written/-transcribed text involves an explicit statement of sentiment while voice modulation and facial features may convey less explicit though perhaps more emotion-laden information. For example, in the debate data (Bucy et al. (2018)) some speakers are more controlled in their speech (as they express aggression through carefully

[2] https://imotions.com/facial-expressions/
[3] https://github.com/TadasBaltrusaitis/OpenFace

planned statements) as opposed to other speakers who have explicit vocal and tonal features marking their aggression.

Algorithm 3 describes the One-Step DCCA algorithm. After applying DCCA once to obtain correlated representations of text ($\bar{\mathbf{v}}_t$) and audio-video ($\bar{\mathbf{v}}_{a,v}$), the input embeddings are concatenated with the correlated embeddings to obtain the final representation of the text and audio-video views as indicated in line 3. The final multi-view embedding $\mathbf{v}_{\text{multi}}$ is obtained by concatenating the final text and audio-video representations.

Algorithm 3 One-Step DCCA

Require: $\mathbf{v}_t, \mathbf{v}_a, \mathbf{v}_v$

1: Initialize $\mathbf{v}_1 = \mathbf{v}_t$, $\mathbf{v}_2 = [\mathbf{v}_a \,|\, \mathbf{v}_v]$.
2: $(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2) \leftarrow \text{DCCA}(\mathbf{v}_1, \mathbf{v}_2)$.
3: $\hat{\mathbf{v}}_t = [\bar{\mathbf{v}}_1 | \, \mathbf{v}_1]$, $\hat{\mathbf{v}}_{a,v} = [\bar{\mathbf{v}}_2 | \, \mathbf{v}_2]$.
4: Return $\mathbf{v}_{\text{multi}} = [\hat{\mathbf{v}}_t | \hat{\mathbf{v}}_{a,t}]$.

Two-Step DCCA

Since we are interested in studying the interplay of audio, video, and text views when learning multi-modal embeddings, in this framework we empirically explore the optimal combination of input views to DCCA. For example, we can fix one input view to be audio. The second view is obtained as the output from a separate DCCA operation that takes as inputs text and video views. The correlated text and video views are then concatenated to form second input view. This way we perform two DCCA steps as suggested by the name of the method.

Algorithm 4 briefly describes the Two-Step DCCA algorithm. This algorithm is the same as Algorithm 3 with the exception of lines 3, where we take the output of the first DCCA step to obtain one input view for the second DCCA step.

Sentiment Classification

Multi-view embeddings obtained from One-Step DCCA and Two-Step DCCA are input to a logistic regression classifier to predict the sentiment label for test data

Algorithm 4 Two-Step DCCA

Require: $\mathbf{v}_t, \mathbf{v}_a, \mathbf{v}_v$

1: Initialize $(\mathbf{v}_1 = \mathbf{v}_t, \mathbf{v}_2 = \mathbf{v}_v)$ or $(\mathbf{v}_1 = \mathbf{v}_v, \mathbf{v}_2 = \mathbf{v}_a)$ or $(\mathbf{v}_1 = \mathbf{v}_t, \mathbf{v}_2 = \mathbf{v}_a)$.
2: $(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2) \leftarrow \text{DCCA}(\mathbf{v}_1, \mathbf{v}_2)$.
3: Initialize $\mathbf{v}'_1 = [\bar{\mathbf{v}}_1 | \bar{\mathbf{v}}_2]$ and $\mathbf{v}'_2 = \mathbf{v}_a$ *or* $\mathbf{v}_t$ *or* $\mathbf{v}_v$.
4: $(\bar{\mathbf{v}}'_1, \bar{\mathbf{v}}'_2) \leftarrow \text{DCCA}(\mathbf{v}'_1, \mathbf{v}'_2)$.
5: $\hat{\mathbf{v}}'_1 = [\bar{\mathbf{v}}'_1 | \, \mathbf{v}'_1]$, $\hat{\mathbf{v}}'_2 = [\bar{\mathbf{v}}'_2 | \, \mathbf{v}'_2]$.
6: Return $\mathbf{v}_{\text{multi}} = [\hat{\mathbf{v}}'_1 | \hat{\mathbf{v}}'_2]$.

sets in Section 5.6.

5.6 Experiments

This section first describes the different test data sets used and the baseline methods that are evaluated against embeddings obtained from One-Step DCCA and Two-Step DCCA. Multi-modal embeddings obtained from DCCA methods are input to a logistic regression classifier and accuracy and F-scores on test data sets are reported as the performance metrics.

Test Data Sets

The following data sets are used for testing,

- **CMU-MOSI:** This is a standard benchmark data set of product reviews curated by 93 users and introduced by (Zadeh et al. (2016)). Reviews are in the form of videos that are segmented into clips. Each clip is assigned a sentiment score between -3 to 3 by five annotators. Sentiment scores are further binarized as 'positive' and 'negative', by assigning all reviews having scores ≥ 0 as 'positive' and all scores < 0 as 'negative'. There are a total of 2198 data points in the classification task. Predetermined splits of training data (1283 points), validation data (229 points) and test data (686 points) are used in our experiments.

- **CMU-MOSEI:** This data set (Zadeh et al. (2018b)) is similar to MOSI and is also annotated at the utterance/clip level. Each utterance is assigned a sentiment score between -3 to 3. Frames scored $(0, 3]$ are labeled as 'positive' and scores between $[-3, 0)$ are labeled as 'negative'. There are a total of 17859 data points available for binary sentiment classification. Data is partitioned into predetermined train (12787 points), validation (3634 points) and test (1438 points) splits. Raw features for CMU-MOSI and CMU-MOSEI are obtained from CMU-Multimodal SDK (Zadeh et al. (2018a)).

- **Debate Emotion:** This data set was curated by (Bucy et al. (2018)) by combining the first and third of the 2016 presidential debates. Data is divided into short videos each of a 10 second duration. Videos that contained multiple speakers in a 10-second duration were not considered, thereby restricting each video to a single speaker. This results in a total of 800 short videos, with a single candidate speaking for 10 seconds. Candidate videos were annotated for 'agression' based on candidate expression on three views: verbal, facial and tonal. An aggression label of 1 was assigned to the video if the candidates expressed anger or aggression in either of the three views. This data set consists of 800 data points, partitioned into train (510 points), validation (90 points) and test (200 points) splits.

Baseline Algorithms

Since the focus of these experiments is to demonstrate i) the combination of three views is better than unimodal feature embeddings in sentiment analysis tasks and ii) study the contribution of various views in multi-modal embeddings used for classification we compare against the following baselines in this work,

- **Unimodal features:** To empirically confirm our hypothesis that three views do better than one, multi-modal embeddings are compared against text, audio and video embeddings when input separately to a logistic regression classifier. Additionally we also compare against bi-modal features obtained by

taking concatenations of audio/video, video/text, text/audio and passing the concatenated features as input to a logistic regression classifier.

- **bc-LSTM+3D-CNN+openSMILE:** This algorithm was introduced by (Poria et al. (2018)) and uses a bidirectional LSTM to encode for text, 3D-CNN and openSMILE to obtain visual and audio embeddings.

- **Generalized Canonical Correlation Analysis:** GCCA (Carroll (1968)) aims to find a correlation subspace in which weighted combinations of all the input views are correlated. Since each view may contribute differently towards a downstream task, GCCA employs a technique to learn weights corresponding to the importance of each view. A discussion detailing the mechanics of the algorithm are beyond the scope of this chapter and we refer readers to (Benton et al. (2016)). Since we explore a Two-Step approach as a potential combination technique for the three views, it is best compared against a technique like GCCA, that learns a weighted combination of all views.

- **Graph Memory Fusion Network:** This algorithm (Zadeh et al. (2018b)) proposes a dynamic fusion graph to analyze the interactions between different modalities at the word level. LSTMs are used to capture information for the whole sequence. Note that this baseline does not directly compare against our method, since it operates at the word level. However, since this algorithm achieves the state-of-the-art on the MOSEI data set, it is included in this work.

Hyperparameters for DCCA: DCCA implementation in this work follows that of (Andrew et al. (2013)) with three fully connected feed-forward layers and a ReLU activation applied to the output of each connected layer. DCCA objective is optimized using RMSProp as in the original implementation. Sizes of connected layers are determined via grid search. A similar technique is used to determine parameters of the Logistic Regression classifier.

Data View	Debate Emotion		CMU-MOSI		CMU-MOSEI	
	Acc	F-score	Acc	F-score	Acc	F-score
Audio	85.0	85.8	44.5	45.0	51.21	51.94
Video	81.0	82.06	44.0	44.5	58.75	59.23
Text	77.5	76.8	78.8	79.17	80.23	83.00
Audio+Video	82.5	83.0	49.0	50.1	62.46	63.03
Audio+Text	85.0	85.3	79.8	79.7	82.88	83.2
Video+Text	85.5	83.2	79.44	79.41	83.05	83.12
Audio+Video+Text (One-Step DCCA)	**93.0**	**93.1**	**80.6**	**80.57**	**83.62**	**83.75**
Audio+Video+Text (GCCA)	88.0	87.9	78.0	77.36	83.02	81.16
Audio+Video+Text (Logistic Reg)	91.0	90.9	79.5	76.6	82.97	83.20
Audio+Video+Text (bc-LSTM+3D-CNN+openSMILE)	N/A	N/A	78.8*	N/A	N/A	N/A
Audio+Video+Text(Graph Memory Fusion Network)	N/A	N/A	N/A*	N/A*	76.9*	77.0*

Table 5.1: This table presents accuracy and F-score for One-Step DCCA and baseline methods on MOSI, MOSEI and Debate Emotion data sets. Best performing algorithm and modality are indicated in boldface. Star marked results correspond to numbers reported in original publication.

Data View	Debate Emotion		CMU-MOSI		CMU-MOSEI	
	Acc	F-score	Acc	F-score	Acc	F-score
$\mathbf{v}_1 = \mathbf{v}_a$,$\mathbf{v}_2 = \mathbf{v}_v$,$\mathbf{v}_2' = \mathbf{v}_t$	**92.5**	**92.57**	**80.17**	**80.32**	**83.57**	**83.71**
$\mathbf{v}_1 = \mathbf{v}_t$,$\mathbf{v}_2 = \mathbf{v}_v$,$\mathbf{v}_2' = \mathbf{v}_a$	92.0	92.05	79.33	79.46	83.23	83.30
$\mathbf{v}_1 = \mathbf{v}_t$,$\mathbf{v}_2 = \mathbf{v}_a$,$\mathbf{v}_2' = \mathbf{v}_v$	91.5	91.34	79.45	79.44	83.19	83.33

Table 5.2: This table presents results from the Two-Step DCCA procedure for all combination of input views. Accuracy of the best performing combination on each data set is represented in boldface.

Experimental Results

Table 5.1 presents accuracy and F-score obtained by baseline methods and One-Step DCCA on the MOSI, MOSEI and Debate Emotion data sets. Results from Table 5.1 indicate that, unsurprisingly, making use of all three views when learning feature embeddings provides the best result. Furthermore, learning combinations in a correlated space using DCCA consistently outperforms other baselines on all three data sets. Note that, the performances reported for the bc-LSTM+3D-CNN+openSMILE baseline and the Graph Memory Fusion Network are as reported in (Poria et al. (2018)) and (Zadeh et al. (2018b))[4]. We do not reproduce code

[4]There is considerable difference in the features used for learning multi-modal embeddings with DCCA and the features used in (Zadeh et al. (2018b)). Comparisons are only between reported

from (Poria et al. (2018)), (Zadeh et al. (2018b)) for evaluation on Debate Emotion dataset. All other baselines were reproduced on the test data sets using parameters reported in their original implementations.

Table 5.2 represents results from the Two-Step DCCA process. From the empirical results it can be noted that i) Two-Step DCCA performs just as well as One-Step DCCA and not better and ii) performing DCCA first with audio and video as input views and then with text as input view to the second DCCA step is the best performing combination. Note that the dimension of the multi-modal embedding learned using DCCA is upper bounded by $\leq \min(d_1, d_2)$ where, d_1 is the dimension of first input view and d_2 is the dimension of the second input view. Since there is a large disparity in the dimensions of the text (768) and audio (74) and video (35) views, performing Two-Step DCCA results in multi-modal embedding that is smaller in size that the multi-modal embedding obtained from One-Step DCCA. This loss in dimensionality may lead to information loss further leading to lower quality multi-modal embeddings. The second result is not too surprising either. Given that the text embeddings are highly optimized when compared to the audio and video embeddings, it is not surprising that learning correlations between audio and video embeddings and then combining them with a text embedding produces the best result.

5.7 Discussions and Conclusions

The key issue is this: which is more important, text, speech, or video? Being able to process all three views of human discourse simultaneously allows consideration of the basic question of the relative contributions of the semantics, the spoken delivery, and the accompanying images. Conventional wisdom would be that the text (the "meaning") of a statement is the most significant. Common sense argues that the spoken word can have an impact – one can usually distinguish an argument from a casual conversation even in an unknown language. Erik Bucy (Grabe and Bucy (2009)) argues that it is the images that can be the most significant aspect in the

performance metrics.

political framing of issues. Our experiments speak to this issue in a concrete way by showing the classification accuracy using two of the views/modes improves on any single view, and that using all three results in the greatest improvement. Digging deeper, the experiments suggest better (and worse) ways of structuring the interactions between the modalities, which we explore by contrasting the ”one-step” and ”two-step approaches”.

At this point, conclusions about the relative importance of the three views should not be taken too quantitatively. First, algorithms such as BERT designed for deriving text-based features are quite sophisticated and have been trained on Wikipedia-sized corpora. In contrast, both audio and video feature extraction has not received nearly the attention that has been given to text. Second, the corpora on which the speech and video have been studied are comparatively small. This means that the results likely underestimate the importance of these two views and accentuate the importance of the text. Third, we have no way of knowing if our chosen classification tasks are representative of other common tasks. Results seem overall analogous across three data sets and two tasks, but this is far from a generic representation. Finally, we have no reason to believe that our method of merging the views is optimal. Indeed, there are many possibilities, and while the trends tend to be similar, there is a lot of variation as different (hyper)parameters are considered and different structures are investigated.

6 APPLICATIONS AND FUTURE WORK

6.1 Overview

This chapter presents results from dictionary based methods for sentiment analysis tasks as well as the applications of DA embeddings in tasks other than sentiment analysis. Brief discussion of results from these tasks is followed by directions for proposed future work.

6.2 Detecting Recovery Problems Just in Time: Application of Automated Linguistic Analysis and Supervised Machine Learning to an Online Substance Abuse Forum.

This work first appeared in (Kornfield et al. (2018)) and explores word count based embedding algorithms to learn efficient text representation for classification tasks. The focus of this work is towards learning classifiers, that can classify emotions such as 'threat' from texts posted on digital intervention treatment forums. Such forums provide a platform for people with substance use disorders (SUDs) such as alcohol addiction or opioid addiction. The A-CHESS data set discussed in chapters 2 and 4 comes from this study.

Dictionary based embedding methods such as Linguistic Inquiry and Word Count (LIWC) (citation) preceded the modern neural network based embedding algorithms. While these algorithms do not scale well, they are particularly efficient in capturing syntax and semantics from text. A method like LIWC develops a rule-based approach that counts occurrences of semantic features such as pronouns, personal pronouns or emotion words expressing emotions such as positive emotion, negative emotion, affect and also considers word count in text when learning a

feature representation for text. Using these rules each piece of text can be mapped into an n dimensional vector. Custom dictionaries can then be developed using similar rules like in LIWC tailored to a particular data set.

For example, for the A-CHESS data set, some custom categories correspond to expressions of alcohol consumption/use, peer support towards addictions etc.

In this work (Kornfield et al. (2018)) me make use of a combination of dictionary approaches like LIWC and count based approaches like Bag of Words (BOW) to learn features for text. While LIWC and BOW features perform similarly when input to a decision tree based classifier, BOW method is much slower. However, empirically it is observed that a concatenation of BOW and LIWC features performs best when input to a decision tree classifier. Using decision tree classifiers also provides an insight into the key 'feature words' from text. For example, some of the most important feature words captured by the BOW approach are 'Drink', 'Depressed', 'Struggle' and 'Frustration' while some of the most important LIWC features are 'Tone', 'Clout', 'Cognitive processing' and 'Insight'.

Such a qualitative analysis formed the basis for methods developed in Chapters 2 and 4.

6.3 Using time series and natural language processing to identify viral moments in the 2016 U.S. Presidential Debate

This work that is to appear in (3^{rd} NLP + CSS Workshop, NAACL 2019) proposes a combination of time series analysis and the domain adaptive embeddings discussed in chapter 2 to study 'media storms', defined as 'an explosive increase in news coverage of a specific item (event or issue) constituting a substantial share of the total news agenda during a certain time.' Since media storms that occur in news media correlate with activity on social media platforms, data from social media platforms like Twitter provide interesting data to be analyzed when modeling media storms.

In this work, presidential debates are studied to understand media storms. This is because, presidential debates fulfill all three conditions of a media storm i) they are explosive (attention to them occurs suddenly, as the presidential debate starts), ii) large (they consume most media attention until the event is over) and ii) can be long-lasting (post-debate spin ensures that coverage of the debate lasts for longer than 24 hours week) Fridkin et al. (2008).

A combination of time series analysis and application of DA embeddings is used explore a set of possible viral moments induced by the debate's candidates. Once a viral moment within a debate is determined via time series analysis of Twitter data, adaptive embeddings are used to determine if the viral moment truly induces a discursive shift.

To do this, data from two minutes before and after an identified viral moment are obtained and defined as two different domains. The DA embedding procedure outlined in chapter 2 is then applied to the 'pre' and 'post' domains and adapted embeddings are obtained. For a vocabulary of words common in both domains, the l_2 distance between the two embeddings are obtained. This technique to measure domain shift is explained in chapter 3. By measuring the words the shifted the most in use two minutes pre and post the word's utterance in the debate discursive shift is quantified.

Debates considered in this work are the 2016 U.S presidential debates. Viral moments are particularly studied for the then candidates Donald Trump and Hilary Clinton. An example of a viral moment caused by Clinton is the statement that Trump "thinks that climate change is a hoax" and an example of a viral moment caused by Trump is the use of the word "braggadocious" regarding his income and competency. From the Clinton viral moment, words about climate change shifted greatly, like 'green', 'climate', 'energy', and 'change'. Other discourse-specific words also had strong discursive shifts, such as 'hoax' and 'China' (words that originated directly from Clinton's statement). For Trump, the words with the greatest l_2-distance difference in the pre- and post-viral moment were related to the topic Trump was discussing. However neither the word 'braggadocious', nor the presumed root word 'brag', appeared on our list (it is likely that the word

n	Word	Δ L2 distance
1	blah	47.95
2	made	41.93
3	fuck	39.47
4	said	38.71
5	**green**	38.06
6	**climate**	37.57
7	**energy**	36.32
8	looks	36.28
9	again	35.19
10	**real**	33.80
11	because	33.71
12	sexist	33.68
13	**change**	33.54
14	**hoax**	33.38
15	important	32.93
16	please	32.21
17	bush	32.07
18	**china**	30.65
19	those	30.48
20	does	29.69

Table 6.1: Words with the greatest l2 distance difference between the pre-viral and post-viral moment for Clinton's Viral Moment 1

n	Word	Δ L2 distance
1	**wrong**	102.62
2	**Iraq**	101.83
3	**should**	73.67
4	take	62.94
5	**China**	57.53
6	there	53.07
7	security	51.86
8	really	51.56
9	talking	45.79
10	**money**	45.33
11	wants	45.02
12	racial	44.28
13	only	41.38
14	plan	41.30
15	even	41.00
16	better	40.14
17	maybe	39.66
18	endorse	38.90
19	lost	36.91
20	**International**	36.15

Table 6.2: Words with the greatest l_2 distance difference between the pre-viral and post-viral moment for Trump's Viral Moment 2

braggadocious did not appear at all in the pre-viral moment). Instead, the discourse shift on social media seemed to center around the foreign policy implications of his statement, which Trump pivoted to immediately following his statement about being a good businessman (this is what he was being 'braggadocious' about). Although Trump did not explicitly mention any countries in his statement, social media discourse focused on countries like China and Iraq (two countries that Trump mentions frequently elsewhere). However, the prevalence of more unrelated words suggests that this potential viral moment did not result in as strong of a discourse shift as Clinton's viral moments. More succinctly put: Trump's statement likely resulted in a spike of attention; however, this shock did not focus attention specifically on Trump's words the way shocks in attention to Clinton did.

Tables 6.1 and 6.2 present a list of the top 20 words that shifted the most for each viral moment.

6.4 Future Work

There are several possible extensions for work centered around learning domain adaptive embeddings. First, chapters 2 and 3 demonstrate that domain adaptive embeddings perform well on quantitative tasks such as sentiment classification on binary and multi-class data sets. However, there is only preliminary exploration towards the qualitative properties of the domain adaptive embeddings.

In addition to measuring 'shift' using l2 distances, one can study the properties of the adaptive embeddings by analyzing the CNN filters used to learn the DA embeddings as described in chapter 3. A similar technique to analyze filters in a transfer learning framework has been performed by (He et al. (2018b))

An important extension for this work is towards developing models that can be trained end-to-end by incorporating losses from the CCA objective and well as the task specific loss function. Current implementations of the CCA objective make use of SVD operations that do not integrate well with a task such as classification. This is because the matrix representations used in the CCA objectives do not directly effect the function variables used to calculate classification losses. To overcome

this, a transformation on the matrices used to obtain the CCA projections needs to be determined. Once such a transformation is obtained, by working with the spectrum of the matrix one can integrate the CCA variables with those of the task specific objective. This way we can perform an end-to-end training of the CCA space as well as the other parameters of the encoder networks used.

REFERENCES

An, Jisun, Haewoon Kwak, and Yong-Yeol Ahn. 2018. Semaxis: A lightweight framework to characterize domain-specific word semantics beyond sentiment. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, 2450–2461. Association for Computational Linguistics.

Andrew, Galen, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *International conference on machine learning*, 1247–1255.

Arora, Raman, and Karen Livescu. 2013. Multi-view cca-based acoustic features for phonetic recognition across speakers and domains. In *2013 ieee international conference on acoustics, speech and signal processing*, 7135–7139. IEEE.

Baltrušaitis, Tadas, Marwa Mahmoud, and Peter Robinson. 2015. Cross-dataset learning and person-specific normalisation for automatic action unit detection. In *2015 11th ieee international conference and workshops on automatic face and gesture recognition (fg)*, vol. 6, 1–6. IEEE.

Baltrušaitis, Tadas, Peter Robinson, and Louis-Philippe Morency. 2016. Openface: an open source facial behavior analysis toolkit. In *2016 ieee winter conference on applications of computer vision (wacv)*, 1–10. IEEE.

Benton, Adrian, Raman Arora, and Mark Dredze. 2016. Learning multiview embeddings of twitter users. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*, vol. 2, 14–19.

Bilenko, Natalia Y, and Jack L Gallant. 2016. Pyrcca: regularized kernel canonical correlation analysis in python and its applications to neuroimaging. *Frontiers in neuroinformatics* 10.

Blitzer, John, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Acl*, vol. 7, 440–447.

Bucy, Erik P, Jordan M Foley, Josephine Lukito, Larisa Doroshenko, Dhavan V Shah, Jon Pevehouse, Chris Wells, and Erik P Bucy. 2018. Performing populism: TrumpâĂŹs transgressive debate style and the dynamics of twitter response. *New Media & Society*.

Carroll, J Douglas. 1968. Generalization of canonical correlation analysis to three or more sets of variables. In *Proceedings of the 76th annual convention of the american psychological association*, vol. 3, 227–228.

Conneau, Alexis, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017a. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Conneau, Alexis, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017b. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, 670–680. Copenhagen, Denmark: Association for Computational Linguistics.

Deerwester, Scott, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391.

Degottex, Gilles, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. 2014. CovarepâĂŤa collaborative voice analysis repository for speech technologies. In *2014 ieee international conference on acoustics, speech and signal processing (icassp)*, 960–964. IEEE.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dumais, Susan T. 2004. Latent semantic analysis. *Annual review of information science and technology* 38(1):188–230.

Dumpala, Sri Harsha, Imran Sheikh, Rupayan Chakraborty, and Sunil Kumar Kopparapu. Audio-visual fusion for sentiment classification using cross-modal autoencoder.

Firth, Joseph, John Torous, Jennifer Nicholas, Rebekah Carney, Simon Rosenbaum, and Jerome Sarris. 2017. Can smartphone mental health interventions reduce symptoms of anxiety? a meta-analysis of randomized controlled trials. *Journal of Affective Disorders*.

Flaxman, Seth, Dino Sejdinovic, John P Cunningham, and Sarah Filippi. 2016. Bayesian learning of kernel embeddings. *arXiv preprint arXiv:1603.02160*.

Fridkin, Kim L, Patrick J Kenney, Sarah Allen Gershon, and Gina Serignese Woodall. 2008. Spinning debates: The impact of the news media's coverage of the final 2004 presidential debate. *The International Journal of Press/Politics* 13(1):29–51.

Friedland, Wells, Wagner Shah, and Abhishek. 2017. The civic state under threat: How social, political and media changes eroded wisconsin's civic culture.

Grabe, Maria Elizabeth, and Erik Page Bucy. 2009. *Image bite politics: News and the visual framing of elections*. Oxford University Press.

Hamilton, William L., Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, 595–605. Austin, Texas: Association for Computational Linguistics.

Hangya, Viktor, Fabienne Braune, Alexander Fraser, and Hinrich Schütze. 2018. Two methods for domain adaptation of bilingual tasks: Delightfully simple and broadly applicable. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, 810–820. Association for Computational Linguistics.

Hardoon, David R, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation* 16(12):2639–2664.

He, Ruidan, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018a. Adaptive semi-supervised learning for cross-domain sentiment classification. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, 3467–3476. Association for Computational Linguistics.

———. 2018b. Adaptive semi-supervised learning for cross-domain sentiment classification. *arXiv preprint arXiv:1809.00530*.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hotelling, Harold. 1992. Relations between two sets of variates. In *Breakthroughs in statistics*, 162–190. Springer.

Hu, Anthony, and Seth Flaxman. 2018. Multimodal sentiment analysis to explore the structure of emotions. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining*, 350–358. ACM.

Jain, Prateek, Praneeth Netrapalli, and Sujay Sanghavi. 2013. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual acm symposium on theory of computing*, 665–674. ACM.

K Sarma, Prathusha, Yingyu Liang, and Bill Sethares. 2018. Domain adapted word embeddings for improved sentiment classification. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 2: Short papers)*, 37–42. Melbourne, Australia: Association for Computational Linguistics.

Kim, Yoon. 2014a. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

———. 2014b. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)*, 1746–1751. Doha, Qatar: Association for Computational Linguistics.

Kornfield, Rachel, Prathusha K Sarma, Dhavan V Shah, Fiona McTavish, Gina Landucci, Klaren Pe-Romashko, and David H Gustafson. 2018. Detecting recovery problems just in time: Application of automated linguistic analysis and supervised machine learning to an online substance abuse forum. *Journal of medical Internet research* 20(6):e10136.

Kotzias, Dimitrios, Misha Denil, Nando De Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, 597–606. ACM.

Labutov, Igor, and Hod Lipson. 2013. Re-embedding words. In *Acl (2)*, 489–493.

Le, Quoc V, and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Icml*, vol. 14, 1188–1196.

Li, Ping, Benjamin Schloss, and D Jake Follmer. 2017. Speaking two âĂIJlanguagesâĂİ in america: A semantic space analysis of how presidential candidates and their supporters represent abstract political concepts differently. *Behavior research methods* 49(5):1668–1685.

Lin, Yi, Yoonkyung Lee, and Grace Wahba. 2002. Support vector machines for classification in nonstandard situations. *Machine learning* 46(1-3):191–202.

Lin, Zhouhan, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Litvin, Erika B, Ana M Abrantes, and Richard A Brown. 2013. Computer and mobile technology-based interventions for substance use disorders: An organizing framework. *Addictive behaviors* 38(3):1747–1756.

Liu, Pengfei, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Liu, Qi, Yue Zhang, and Jiangming Liu. 2018. Learning domain representation for multi-domain sentiment classification. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)*, 541–550. Association for Computational Linguistics.

Mairal, Julien, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R Bach. 2009. Supervised dictionary learning. In *Advances in neural information processing systems*, 1033–1040.

McAuley, Julian, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international acm sigir conference on research and development in information retrieval*, 43–52. ACM.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomas, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Mohr, David C, Michelle Nicole Burns, Stephen M Schueller, Gregory Clarke, and Michael Klinkman. 2013. Behavioral intervention technologies: evidence review and recommendations for future research in mental health. *General hospital psychiatry* 35(4):332–338.

Moore, Brent A, Tera Fazzino, Brian Garnet, Christopher J Cutter, and Declan T Barry. 2011. Computer-based interventions for drug use disorders: a systematic review. *Journal of substance abuse treatment* 40(3):215–223.

Nam, Hyeonseob, and Bohyung Han. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 4293–4302.

Naslund, John A, Lisa A Marsch, Gregory J McHugo, and Stephen J Bartels. 2015. Emerging mhealth and ehealth interventions for serious mental illness: a review of the literature. *Journal of mental health* 24(5):321–332.

Pang, Bo, and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, 115–124. Association for Computational Linguistics.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. 2014a. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)*, 1532–1543.

———. 2014b. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)*, 1532–1543. Doha, Qatar: Association for Computational Linguistics.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)*, 2227–2237. Association for Computational Linguistics.

Peters, Matthew E, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Poria, Soujanya, Navonil Majumder, Devamanyu Hazarika, Erik Cambria, Alexander Gelbukh, and Amir Hussain. 2018. Multimodal sentiment analysis: Addressing key issues and setting up the baselines. *IEEE Intelligent Systems* 33(6):17–25.

Qian, Qiao, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2017. Linguistically regularized lstm for sentiment classification. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, 1679–1689. Association for Computational Linguistics.

Quanbeck, Andrew, Ming-Yuan Chih, Andrew Isham, Roberta Johnson, and David Gustafson. 2014. Mobile delivery of treatment for alcohol use disorders: A review of the literature. *Alcohol research: current reviews* 36(1):111.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Rakhlin, Alexander, Ohad Shamir, and Karthik Sridharan. 2011. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*.

Rastogi, Pushpendre, Benjamin Van Durme, and Raman Arora. 2015. Multiview lsa: Representation learning via generalized cca. In *Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 556–566.

Sarma, Prathusha K, and William Sethares. 2018. Simple algorithms for sentiment analysis on sentiment rich, data poor domains. In *Proceedings of the 27th international conference on computational linguistics*, 3424–3435.

Shang, Jingbo, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2054–2064. Association for Computational Linguistics.

Socher, Richard, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, 151–161. Association for Computational Linguistics.

Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Turney, Peter D, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37(1):141–188.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Yin, Wenpeng, and Hinrich Schütze. 2016. Learning word meta-embeddings. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, vol. 1, 1351–1360.

Zadeh, Amir, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. 2018a. Multi-attention recurrent network for human communication comprehension. *arXiv preprint arXiv:1802.00923*.

Zadeh, Amir, Paul Pu Liang, Jon Vanbriesen, Soujanya Poria, Erik Cambria, Minghai Chen, and Louis-Philippe Morency. 2018b. Multimodal language analysis in the wild: CMU-MOSEI dataset and interpretable dynamic fusion graph. In *Association for computational linguistics (acl)*.

Zadeh, Amir, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016. Mosi: multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv:1606.06259*.

Zeng, Jiali, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. 2018. Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, 447–457. Association for Computational Linguistics.