Domain Adapted Word Embeddings for Improved Sentiment Classification

Anonymous ACL submission

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 paper proposes a method to combine the breadth of generic embeddings with the specificity of domain specific embeddings. These new embeddings, called *Domain Adapted* (DA) word embeddings, are formed by aligning corresponding word vectors using Canonical Correlation Analysis (CCA) or the related nonlinear Kernel CCA. Evaluation results on sentiment classification tasks show that the DA embeddings significantly outperform both generic and DS embeddings when used as input features to standard or state-of-the-art sentence encoding algorithms for classification.

1 Introduction

Generic word embeddings such as Glove and word2vec (Pennington et al., 2014; Mikolov et al., 2013) which are pre-trained on large sets of raw text, have demonstrated remarkable success when used as features to a supervised learner in various applications such as the sentiment classification of text documents. There are, however, many applications with domain specific vocabularies and relatively small amounts of data. The performance of word embedding approaches in such applications is limited, since word embeddings pre-trained on generic corpora do not capture domain specific semantics/knowledge, while embeddings trained 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), which contains messages on discussion forums for people with substance addictions. These forums are part of a mobile health intervention treatments that encourages participants to engage in sobriety-related discussions. The data is both domain specific and limited in size. Other examples include reviews of restaurants and movies, discussions by special interest groups, and political surveys. In general they are common in fields like business, social media, and social sciences.

Such data sets present significant challenges for algorithms based on word embeddings. First, the data is on specific topics and has a very different distribution from generic corpora, so pre-trained generic word embeddings such as those trained on Common Crawl or Wikipedia are unlikely to yield accurate results in downstream tasks. When performing sentiment classification using pre-trained word embeddings, differences in domains of training and test data sets limit the applicability of the embedding algorithm. 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 train a word embedding. The SUD data set consists of a few hundred people and only a fraction of these are active (Firth et al., 2017) and (Naslund et al., 2015). This results in a small data set of text messages available for analysis. Furthermore, the content is generated spontaneously on a day to day basis, and language use is informal and unstructured. Running the generic word embedding constructions algorithms on such a data set leads to very noisy outputs that are not suitable as input for downstream applications like sentiment classification. 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.

This paper proposes a method for obtaining high quality word embeddings that capture domain specific semantics and are suitable for tasks on the specific domain. The new embeddings are obtained by combining generic embeddings and Domain Specific (DS) embeddings. Generic embeddings are trained on large corpora and do not capture domain specific semantics, while DS embeddings are obtained from the domain specific data set via algorithms such as Latent Semantic Analysis (LSA) or other embedding methods and can be quite noisy. These two sets of embeddings are combined using a linear CCA (Hotelling, 1936) or a nonlinear kernel CCA (KCCA) (Hardoon et al., 2004). They are projected along the directions of maximum correlation, and a new Domain Adapted (DA) embedding is formed from vectors that are optimally located between the corresponding projected word vectors. The DA embeddings are then evaluated in a sentiment classification setting. Empirically, it is shown that the combined DA embeddings improve substantially over the generic embeddings and over the noisy DS embeddings.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 briefly introduces the CCA/KCCA and details the procedure used to obtain the DA word embeddings. Section 4 presents results from sentiment classification tasks on benchmark data sets using a standard classification baseline as well as using the state-of-the-art sentence encoding algorithm. Section 5 concludes this work.

2 Related Work

This work is related to three areas of research which are outlined below.

CCA based word embeddings and applications in multilingual correlation. In (Dhillon et al., 2012), a Two Step CCA algorithm is used to learn word embeddings from a one hot encoding representation of words in a given vocabulary. CCA has been used to learn multilingual word embeddings (Faruqui and Dyer, 2014) from words aligned in text across different languages. Building on this work, (Lu et al., 2015) developed a deep CCA, i.e., a nonlinear CCA implemented via deep learning, to learn multilingual word embeddings. In both these algorithms, word embeddings are learned for words and their translations across multiple languages such as English-German or English-French, separately via a LSA based approach. Embeddings in the two different languages are then projected onto the best k correlated dimensions via CCA.

Recently, (Gouws et al., 2015) propose a neural network based model that learns across multiple languages without the need for word alignment. This algorithm jointly optimizes learning of monolingual embeddings via an objective similar to (Mikolov et al., 2013), along with a cross lingual alignment task. Recently, CCA has been applied to perform crosslingual entity linking tasks (Tsai and Roth, 2016). While the use of CCA to improve word embeddings in a multilingual setting has been explored, this work uses CCA/KCCA to improve performance of monolingual word embeddings across data sets in different application domains/contexts in downstream sentiment classification tasks.

Domain Adaptation with CCA. The idea of using word embeddings across different domains has been explored by (Luo et al., 2014) where word embeddings are learned independently from two large corpora and then combined via a neural network. This is different from our approach where the CCA-based approach is used to exploit cooccurrences and context information in the domain specific data set along with linear properties of the generic word embedding. More recently, (Yin and Schütze, 2016) proposes an ensemble approach of combining word embeddings learned via different embedding algorithms across different data sets.

Several methods are proposed, e.g., concatenating word vectors from multiple embeddings and then performing dimension reduction using an SVD. The ensemble embeddings are then evaluated on several intrinsic tasks such as word similarities and analogies. Our work focuses on adapting the word vectors to incorporate domain specific knowledge which are important for the extrinsic objective of sentiment classification. The method proposed here of combining word embeddings is also different from theirs.

Some other work (e.g., (Blitzer et al.,

2011), (Anoop et al., 2015) and (Mehrkanoon and Suykens, 2017)) explores CCA based dimensionality reduction techniques for domain adaptation in problems with multimodal data, but these do not directly address natural language data.

Transfer Learning using Sentence Embeddings. The idea of training on a large corpus and testing on a different yet related data set has been successfully explored via transfer learning in computer vision applications (Taigman et al., 2014; Sharif Razavian et al., 2014; Antol et al., 2015). A similar idea has been explored to solve problems in NLP applications via sentence level embeddings with and without composition of word embeddings. An unsupervised algorithm such as skip-thought (Kiros et al., 2015) that adapts the word level skip-gram model by (Mikolov et al., 2013) to sentence level embeddings has demonstrated success in transfer learning tasks. Similarly, (Hill et al., 2016) compare task-specific sentence embeddings to supervised methods for applications on machine translation data.

However, these supervised models fail to perform as well as an unsupervised Skip-Net. The current state-of-the art in sentence embedding algorithms is InferSent (Conneau et al., 2017), which learns a sentence embedding via an encoder trained on the Stanford Natural Language Inference data set. It has demonstrated success in many transfer tasks. While domain adaptation is not the focus of these algorithms, the underlying idea of training a model on one data set/task and testing on a different data set/task is relevant to the theme of this paper. In fact, our experiments demonstrate that our domain adapted embeddings combined with the InferSent architecture can significantly improve over generic embeddings combined with InferSent in the sentiment classification task.

3 Domain Adapted Word Embeddings

Training word embeddings directly on small data sets leads to noisy outputs while embeddings from generic corpora fail to capture specific local meanings within the domain. Here we combine DS and generic embeddings using CCA or kernel CCA (KCCA) which projects corresponding word vectors along the directions of maximum correlation.

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., running 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., running the 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

$$\begin{aligned} \bar{w}_{i,DS} &= \mathbf{w}_{i,DS}\, \phi_{DS} \\ \bar{w}_{i,G} &= \mathbf{w}_{i,G}\, \phi_G. \end{aligned} \tag{1}$$

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

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

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 $\hat{\mathbf{w}}_{i,DA} = \alpha \bar{\mathbf{w}}_{i,DS} + \beta \bar{\mathbf{w}}_{i,G}$, where the parameters α and β can be obtained by solving the following optimization

$$\begin{aligned} \min_{\alpha,\beta} \|\bar{\mathbf{w}}_{i,DS} - (\alpha \bar{\mathbf{w}}_{i,DS} + \beta \bar{\mathbf{w}}_{i,G})\|_2^2 + \\ \|\bar{\mathbf{w}}_{i,G} - (\alpha \bar{\mathbf{w}}_{i,DS} + \beta \bar{\mathbf{w}}_{i,G})\|_2^2. \end{aligned} \tag{3}$$

Solving (3) gives a weighted combination with $\alpha = \beta = \frac{1}{2}$, i.e., the new vector is equal to the average of the two projections:

$$\hat{\mathbf{w}}_{i,DA} = \frac{1}{2}\bar{\mathbf{w}}_{i,DS} + \frac{1}{2}\bar{\mathbf{w}}_{i,G}. \tag{4}$$

Figure 1 illustrates the CCA procedure used to obtain DA word embeddings from the generic and the DS embeddings.

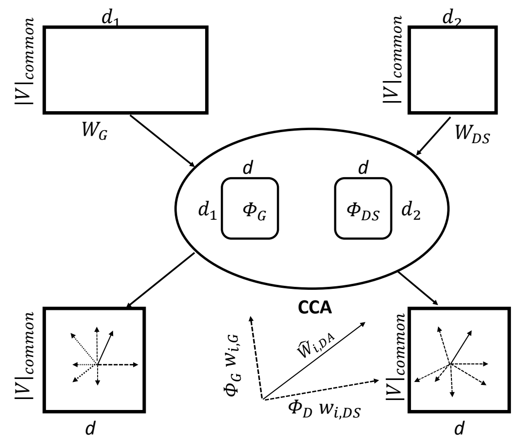

Figure 1: CCA projected vectors from DS and generic word embeddings.

Because of its linear structure, the CCA in (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 the reference for the details.

4 Experimental Evaluation

Comparisons between LSA-based domain specific, the large-scale generic, and the CCA/KCCA Domain Adapted embeddings are carried out on sentiment classification tasks on four data sets. On extrinsic tasks such as sentiment classification, the DA embeddings are expected to generally outperform the others. This is because the DA embeddings have access to both the domain specific semantics via algorithms such as LSA and to generic structure of the large vocabulary via the generic embedding. This is supported by the results reported in Section 4.3.

Furthermore, we demonstrate that the choice of the algorithm for computing DS embedding is also important for obtaining improved DA embeddings. In particular, LSA is a better choice than GloVe or word2vec. To illustrate this point, in Section 4.4, the word2vec(skip-gram) algorithm is used to obtain DS embeddings from domain specific data sets. These embeddings are combined with pre-trained generic word2vec embeddings to get DA embeddings. The results show that, owing to the sizes of the data sets, the DA embeddings obtained via LSA DS do better than DA embeddings obtained via word2vec DS embeddings.

4.1 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

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

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).

4.2 Word Vectors

The following domain specific word embeddings are used:

- **LSA.** Within this framework, a bag-of-words approach is used to build a matrix $\mathbf{C} \in \mathbb{R}^{n \times |V_{DS}|}$ of co-occurrence counts. Then spectral decomposition (SVD) of the co-occurrence counts matrix is performed to obtain word embeddings

$$\mathbf{C} = \mathbf{U}\,\Psi\,\mathbf{V}^{\top}$$

and the left singular vectors corresponding to the d largest singular values are retained to form the DS word embeddings matrix $\mathbf{W}_{DS} \in \mathbb{R}^{|V_{DS}| \times d}$ on the domain data set:

$$\mathbf{W}_{DS} = (\mathbf{V}_d\,\Psi_d).$$

- **word2vec.** In Section 4.4, DS embeddings from the domain specific data sets are obtained by the word2vec model in gensim[2], and compared to those obtained by LSA.

SVD Baseline: In addition to using individual DS and generic embeddings as baselines, a SVD based baseline as used in (Yin and Schütze, 2016) is also used. In this approach, DS and generic embeddings are concatenated to form a word embedding matrix (CONC) on which SVD is performed. Singular vectors projected onto the d largest singular values are used as baseline word embeddings. Note that experimental results of (Yin and Schütze, 2016) show that the SVD based method is more efficient than CONC, hence we consider the SVD baseline alone.

The following generic embeddings are used:

[2] https://radimrehurek.com/gensim/

- **GloVe**. These Word vectors are obtained from the GloVe algorithm and are downloaded online[3]. The experiments used vectors obtained from the Wikipedia corpus (denoted as GloVe) as well as the common crawl embeddings (denoted as GloVe-common crawl) are used.

- **word2vec**. These word vectors are obtained from the word2vec (skip-gram) construction algorithm and are downloaded online[4].

4.3 Sentiment Classification

For sentiment classification, first a sentence encoding method is used to obtained a vector representation of the input sentence (or document), and then the sentence vector is fed into a logistic regression classifier. Two sentence encoding methods are used: a standard one using weighted sum of word embeddings in the sentences, and a sophisticated sentence encoding algorithm InferSent (Conneau et al., 2017). As far as we know, InferSent is the state-of-the-art for sentence encoding. The two methods thus provide a more thorough evaluations of the DA embeddings.

Since the experiments involve an unbalanced data set, average precision, f-score and AUC are reported for all four data sets. Note that the RNTN baseline does not enable calculation of classification probabilities and so AUC scores using this baseline are not reported. Class imbalance is accounted for by penalizing the cost of misclassifying the minor class. Class weights are determined by following the heuristic suggested in (Lin et al., 2002).

4.3.1 Weighted Average Sentence Encoding

In this approach, sentences (or text documents) are expressed as the weighted sum of all the words embeddings present in the document. Weights used are term frequency counts that are obtained while building the LSA. The encoding v is

$$v = \gamma^{\top}\,\mathbf{W}$$

where $\gamma \in \mathbb{R}^{|V|}$ represents the weights for all the words in the sentence/document, and $\mathbf{W}$ is a matrix of word embeddings. A logistic regression classifier is then trained on the training data and

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

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

Data Set	Embedding	Avg Precision	STD	Avg F-score	STD	Avg AUC	STD
	KCCA-DA($\mathbf{W}_G$ = GloVe)	0.8536	0.0288	0.8189	0.0288	0.8257	0.0139
	CCA-DA($\mathbf{W}_G$ = GloVe)	0.8369	0.0476	0.7948	0.0245	0.8033	0.0295
	KCCA-DA($\mathbf{W}_G$ = word2vec)	0.8745	0.0127	0.8336	0.0128	0.8410	0.0095
	CCA-DA($\mathbf{W}_G$ = word2vec)	0.8452	0.0235	0.8002	0.0266	0.8104	0.0217
	KCCA-DA($\mathbf{W}_G$ = common crawl)	**0.8811**	0.0307	**0.8535**	0.0272	**0.8580**	0.0243
	CCA-DA($\mathbf{W}_G$ = common crawl)	0.8622	0.0352	0.8435	0.0245	0.8465	0.0228
Yelp	SVD($\mathbf{W}_G$ = GloVe)	0.8014	0.0264	0.7850	0.0306	0.7892	0.0279
	SVD($\mathbf{W}_G$ = common crawl)	0.8511	0.0232	0.8351	0.0229	0.8380	0.0206
	SVD($\mathbf{W}_G$ = word2vec)	0.8420	0.0371	0.8039	0.0371	0.8083	0.0390
	GloVe	0.7713	0.0427	0.7232	0.0799	0.7417	0.0509
	GloVe-common crawl	0.8210	0.0359	0.7674	0.0346	0.7817	0.0271
	word2vec	0.8280	0.0350	0.7828	0.0350	0.7935	0.0318
	LSA	0.7536	0.0542	0.7117	0.0431	0.7257	0.0432
	KCCA-DA($\mathbf{W}_G$ = GloVe)	0.8630	0.0191	0.8300	0.0298	0.8339	0.0320
	CCA-DA($\mathbf{W}_G$ = GloVe)	0.8468	0.0249	0.8227	0.0222	0.8278	0.0174
	KCCA-DA($\mathbf{W}_G$ = word2vec)	0.8709	0.0186	0.8263	0.0269	0.8350	0.0208
	CCA-DA($\mathbf{W}_G$ = word2vec)	0.8480	0.0158	0.8142	0.0190	0.8212	0.0132
	KCCA-DA($\mathbf{W}_G$ = common crawl)	**0.8973**	0.0247	**0.8547**	0.0242	**0.8556**	0.0263
	CCA-DA($\mathbf{W}_G$ = common crawl)	0.8567	0.0239	0.8383	0.0235	0.8421	0.0211
Amazon	SVD($\mathbf{W}_G$ = GloVe)	0.8236	0.0204	0.8130	0.0351	0.8151	0.0251
	SVD($\mathbf{W}_G$ = common crawl)	0.8728	0.0297	0.8617	0.0254	0.8642	0.0209
	SVD($\mathbf{W}_G$ = word2vec)	0.8493	0.0162	0.7781	0.0233	0.7952	0.0175
	GloVe	0.8158	0.0250	0.7762	0.0271	0.7872	0.0279
	GloVe-common crawl	0.7991	0.0278	0.8163	0.0283	0.8146	0.0265
	word2vec	0.8455	0.0195	0.8052	0.0258	0.8145	0.0202
	LSA	0.8265	0.0445	0.7392	0.0386	0.7640	0.0326
	KCCA-DA($\mathbf{W}_G$ = GloVe)	0.7384	0.0139	0.7307	0.0362	0.7317	0.0248
	CCA-DA($\mathbf{W}_G$ = GloVe)	0.7335	0.0200	0.7300	0.0328	0.7306	0.0203
	KCCA-DA($\mathbf{W}_G$ = word2vec)	**0.8236**	0.0444	**0.7895**	0.0279	**0.7966**	0.0269
	CCA-DA($\mathbf{W}_G$ = word2vec)	0.8066	0.0455	0.7595	0.0457	0.7723	0.0389
	KCCA-DA($\mathbf{W}_G$ = common crawl)	0.5450	0.0254	0.5303	0.0355	0.5391	0.0209
	CCA-DA($\mathbf{W}_G$ = common crawl)	0.5408	0.0203	0.5442	0.0295	0.5490	0.0210
IMDB	SVD($\mathbf{W}_G$ = GloVe)	0.7325	0.0317	0.7455	0.0322	0.7302	0.0474
	SVD($\mathbf{W}_G$ = common crawl)	0.5387	0.0222	0.5177	0.0583	0.5354	0.0196
	SVD($\mathbf{W}_G$ = word2vec)	0.7828	0.0327	0.7767	0.0370	0.7455	0.0295
	GloVe	0.6444	0.0262	0.6518	0.0355	0.6462	0.0266
	GloVe-common crawl	0.5053	0.0189	0.6239	0.0355	0.4996	0.0232
	word2vec	0.7892	0.0373	0.7488	0.0315	0.7560	0.0246
	LSA	0.6792	0.0173	0.6979	0.0533	0.6971	0.0384
	KCCA-DA($\mathbf{W}_G$ = GloVe)	0.3207	0.0132	0.3932	0.0257	0.6596	0.0139
	CCA-DA($\mathbf{W}_G$ = GloVe)	0.327	0.0157	0.3548	0.0428	0.6215	0.0295
	KCCA-DA($\mathbf{W}_G$ = word2vec)	0.3345	0.0134	0.3981	0.0106	0.6592	0.0068
	CCA-DA($\mathbf{W}_G$ = word2vec)	0.3306	0.0320	0.3402	0.0110	0.6091	0.0090
	KCCA-DA($\mathbf{W}_G$ = common crawl)	**0.3638**	0.0123	**0.3471**	0.0486	**0.6136**	0.0261
	CCA-DA($\mathbf{W}_G$ = common crawl)	0.3211	0.0299	0.3685	0.0445	0.6299	0.0316
A-CHESS	SVD($\mathbf{W}_G$ = GloVe)	0.2727	0.0293	0.3445	0.0308	0.6159	0.0239
	SVD($\mathbf{W}_G$ = common crawl)	0.2984	0.0235	0.3632	0.0332	0.6294	0.0110
	SVD($\mathbf{W}_G$ = word2vec)	0.2809	0.0194	0.3506	0.0147	0.6213	0.0262
	GloVe	0.3082	0.0206	0.3367	0.0342	0.6080	0.0231
	GloVe-common crawl	0.3813	0.0089	0.2745	0.0314	0.5749	0.0128
	word2vec	0.3267	0.0299	0.3172	0.0169	0.5964	0.0050
	LSA	0.2742	0.0162	0.3438	0.0237	0.6156	0.0195

Table 1: This table shows results from a standard sentiment classification task on all four data sets. Metrics reported are average Precision, F-score and AUC and the corresponding standard deviations (STD). Best results for each data set are highlighted in boldface.

Data Set	Embedding	Avg Precision	STD	Avg F-score	STD	Avg AUC	STD
	InferSent(common crawl)	0.8647	0.0197	0.8351	0.0264	0.8383	0.0221
	InferSent(retrained-AdaptGloVe(KCCA))	**0.9106**	0.0080	**0.8866**	0.0246	**0.8876**	0.0244
Yelp	InferSent(retrained-AdaptGloVe(CCA))	0.8626	0.0149	0.8261	0.0113	0.8399	0.0086
	InferSent(retrained-SVD)	0.8553	0.0212	0.8490	0.0178	0.8496	0.0158
	RNTN	0.8311	0.0111	-	-	-	-
	InferSent(common crawl)	0.8793	0.0270	0.8241	0.0333	0.8324	0.0280
	InferSent(retrained-AdaptGloVe(KCCA))	**0.9056**	0.0215	**0.8652**	0.0200	**0.8674**	0.0190
Amazon	InferSent(retrained-AdaptGloVe(CCA))	0.8712	0.0266	0.8318	0.0227	0.8378	0.0215
	InferSent(retrained-SVD)	0.8573	0.0196	0.8519	0.0248	0.8517	0.0266
	RNTN	0.8284	0.0067	-	-	-	-
	InferSent(common crawl)	0.5402	0.0321	0.5303	0.0528	0.5301	0.0205
	InferSent(retrained-AdaptGloVe(KCCA))	0.5976	0.0737	**0.5326**	0.0618	0.5646	0.0342
IMDB	InferSent(retrained-AdaptGloVe(CCA))	0.5362	0.0166	0.5062	0.0514	**0.5875**	0.0379
	InferSent(retrained-SVD)	0.5275	0.0231	0.5305	0.0602	0.5354	0.0258
	RNTN	**0.8088**	0.0070	-	-	-	-
	InferSent(common crawl)	0.5221	0.0510	**0.5526**	0.0564	**0.7428**	0.0361
	InferSent(retrained-AdaptGloVe(KCCA))	**0.5537**	0.0550	0.5067	0.0502	0.6989	0.0317
A-CHESS	InferSent(retrained-AdaptGloVe(CCA))	0.5434	0.0368	0.4876	0.0299	0.6878	0.0241
	InferSent(retrained-SVD)	0.4041	0.0424	0.4475	0.0552	0.6813	0.0387
	RNTN	-	-	-	-	-	-

Table 2: This table shows results obtained by using embeddings from the InferSent encoder in a sentiment classification task. Metrics reported are average Precision, F-score and AUC along with the corresponding standard deviations (STD). Best results for each data set are highlighted in boldface.

used to predict the sentiment labels on the test data sets.

Table 1 shows classification results. Using DA embeddings obtained from either CCA or KCCA consistently improves over the corresponding generic or DS embeddings. In general, KCCA leads to better results than CCA. DA embeddings obtained by KCCA on generic GloVe common crawl embeddings give the best performance on all data sets, except for the IMDB data set, on which the best is KCCA with word2vec. Note that the results suggest that the GloVe common crawl vectors are not suitable for this data set while word2vec vectors are much better (which is also supported by InferSent results in the next subsection). Even in this case, our method still improves the generic embeddings.

4.3.2 InferSent Sentence Encoding

First, the pre-trained encoder[5] initialized with GloVe common crawl embeddings is used to obtain vector representations of the input data. Embeddings of the input text data are then used for classification using a logistic regression classifier. This is denoted as InferSent(common crawl).

Second, 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. This is denoted as AdaptGloVe. Vector representations of sentences/documents obtained from the retrained encoder are then used to perform sentiment classification and compared against results obtained from the pre-trained encoder.

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.

Table 2 shows results from sentiment classification tasks using embeddings obtained from the InferSent encoder. The KCCA DA embeddings are still the best on all data except IMDB. On the

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

Word embedding	Dimension
GloVe	100
word2vec	300
LSA	70
CCA-DA	68
KCCA-DA	68
GloVe common crawl	300
AdaptGloVe	300

Table 3: This table presents the average dimensions of LSA, generic and DA word embeddings.

IMDB data set embeddings obtained from the InferSent encoder do not do well even when initialized with AdaptGloVe embeddings. A possible explanation for this could be that the GloVe common crawl vectors are not suitable for this data set. This explanation is consistent even across results from embeddings obtained after retraining the InferSent encoder with the AdaptGloVe embeddings, and consistent with the results in the previous subsection.

Finally, comparing the results in Table 1 and Table 2, it is observed that RNTN is outperformed even by the standard weighted sum with DA embeddings. This demonstrates the effectiveness of our method.

4.3.3 Hyper-parameters

Dimensions of CCA and KCCA projections. Using both KCCA and CCA, generic embeddings and DS embeddings are projected onto their d largest correlated dimensions. By construction, $d \leq \min(d_1, d_2)$, and so the best d for each data set is obtained via 10 fold cross validation on the sentiment classification task. Table 3 provides dimensions of all word embeddings considered. Note that for LSA and DA, average word embedding dimension across all four data sets are reported. Generic word embeddings such as GloVe and word2vec are of fixed dimensions across all four data sets. Since InferSent is initialized with GloVe common crawl embeddings of dimension 300, AdaptGloVe is also of dimension 300.

Kernel parameter estimation. Parameter σ of the Gaussian kernel for use in KCCA is obtained empirically from the data. Pairwise distances between data points mapped by the kernel function are calculated. The median (μ) of these distances is then set as σ. A comprehensive survey detailing the origins of this metric is provided in (Flaxman et al., 2016). Typically $\sigma = \mu$ or $\sigma = 2\mu$. In this section both values are considered for σ and results with the best performing σ are reported. 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).

4.4 Effect of Different DS Embedding Algorithms

The standard sentiment classification task on the Yelp and Amazon data sets is repeated with DS embeddings obtained from the word2vec algorithm.

Table 4 shows the results for DA embeddings from word2vec DS embeddings, generic word2vec embeddings, and the DS word2vec embeddings. Still, KCCA DA embeddings perform better than the DS and the generic embeddings. Comparing the results in Table 1, it is observed that DA embeddings from LSA DS embeddings are better than those from word2vec DS embeddings. These results reinforce the facts that 1) word embedding algorithms optimized by training on large scale corpora such as the Wikipedia corpus do not perform well on small sized data sets and 2) exploiting the specifics of the domain of the data set is vital to improve the performance of generic embeddings, and pectral decomposition algorithms such as the LSA do better in capturing the specifics of small sized data sets.

5 Conclusions and Future Work

In this work, generic word embeddings are combined with domain specific embeddings by projecting along directions of maximum correlation. The resulting domain adapted embeddings, evaluated on sentiment classification tasks from four different data sets show that the DA embeddings outperform both the generic and the DS word embeddings in a standard classification framework.

Furthermore, DA embeddings nearly match the performance of neural network based embeddings algorithms such as InferSent. Initializing InferSent with DA embeddings further improves the performance of embeddings obtained via 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 these algorithms with embeddings customized to a particular domain or data set further improves the performance of the algorithms. Future work will explore effectiveness of using our approach in other downstream applications.

References

KR Anoop, Ramanathan Subramanian, Vassilios Vonikakis, KR Ramakrishnan, and Stefan Winkler. 2015. On the utility of canonical correlation analysis for domain adaptation in multi-view headpose estimation. In *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, pages 4708–4712.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2425–2433.

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

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

John Blitzer, Sham Kakade, and Dean Foster. 2011. Domain adaptation with coupled subspaces. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. pages 173–181.

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

Paramveer Dhillon, Jordan Rodu, Dean Foster, and Lyle Ungar. 2012. Two step cca: A new spectral method for estimating vector models of words. *arXiv preprint arXiv:1206.6403* .

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. Association for Computational Linguistics.

Joseph Firth, 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* .

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

Data Set	Embedding	Avg Precision	STD	Avg F-score	STD	Avg AUC	STD
Yelp	KCCA-DA($\mathbf{W}_G$ = word2vec(Skip-Gram))	**0.8369**	0.0351	**0.7899**	0.0424	**0.8003**	0.0375
	CCA-DA($\mathbf{W}_G$ = word2vec(Skip-Gram))	0.7809	0.0176	0.7604	0.0172	0.7666	0.0153
	word2vec ($\mathbf{W}_G$)	0.8280	0.0350	0.7828	0.0350	0.7935	0.0318
	word2vec ($\mathbf{W}_{DS}$)	0.7308	0.0228	0.7097	0.0244	0.7176	0.0212
Amazon	KCCA-DA($\mathbf{W}_G$ = word2vec(Skip-Gram))	**0.8568**	0.0323	0.8123	0.0326	**0.8220**	0.0298
	CCA-DA($\mathbf{W}_G$ = word2vec(Skip-Gram))	0.8350	0.0344	**0.8131**	0.0404	0.8186	0.0371
	word2vec ($\mathbf{W}_G$)	0.8455	0.0195	0.8052	0.0258	0.8145	0.0202
	word2vec ($\mathbf{W}_{DS}$)	0.7420	0.0588	0.7249	0.0501	0.7311	0.0483

Table 4: This table shows results from a standard sentiment classification the Yelp and Amazon data sets using word2vec DS embeddings. Metrics reported are average Precision, F-score and AUC and standard deviation. Best results for each data set are highlighted in boldface.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. pages 748–756.

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

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483* .

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika* 28(3/4):321–377.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Dimitrios Kotzias, 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*. ACM, pages 597–606.

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

Erika B Litvin, 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.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *HLT-NAACL*. pages 250–256.

Yong Luo, Jian Tang, Jun Yan, Chao Xu, and Zheng Chen. 2014. Pre-trained multi-view word embedding using two-side neural network. In *AAAI*. pages 1982–1988.

Siamak Mehrkanoon and Johan AK Suykens. 2017. Regularized semipaired kernel cca for domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems* .

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

John A Naslund, 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.

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

Andrew Quanbeck, 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.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. pages 806–813.

Richard Socher, 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*. pages 1631–1642.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 1701–1708.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *HLT-NAACL*. pages 589–598.

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

Wenpeng Yin 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)*. volume 1, pages 1351–1360.

Liang-Chih Yu, Jin Wang, K Robert Lai, and Xuejie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 534–539.