Class-specific Word Embedding through Linear Compositionality

Sicong Kuang Brian D. Davison
Lehigh University, Bethlehem PA, USA
{sik211, davison} @cse.lehigh.edu

ABSTRACT

English linguist John Rupert Firth has a famous saying "you shall know a word by the company it keeps." Many recent neural language models such as Word2Vec are based on this assumption. This assumption generally holds. This assumption fails when it comes to the problem of words with (almost) syntactic equivalence and contrasting polarity semantically. For example, "this is a good guy" and "this is a bad guy". Without additional data, based on this assumption a machine learns that "good" is equal to "bad" because they are seen in the same context.

Another problem is related to words with multiple meanings called polysemy. In current popular approaches to learn word representations, such as Word2Vec, words with multiple senses are overwhelmed by the words' general meaning, such as "blue", "apple" and "nuts", each of which has multiple meanings. We observe that this problem can be better resolved when the class information or label of the sentence is presented. For example, in a disaster related classification task, when a sentence is labeled as "hurricane related", the word "water" in the sentence is more likely to be correctly interpreted as rain and flood; when a sentence is labeled as "hurricane-unrelated", the word "water" can be interpreted as its common general meaning.

We propose three approaches to train class-specific embeddings to encode class information by utilizing the linear compositionality property of word embeddings. We present a general framework consisting of a pair of convolutional neural networks for text classification tasks. We evaluate our approach and framework on topic classification of a disaster-focused Twitter dataset and a benchmark Twitter sentiment classification dataset from SemEval 2013. Our results show a potential relative accuracy improvement of more than 5% over a recent baseline.

CCS Concepts

•**Information systems** → *Information extraction;*

DOI: 10.1145/1235

Keywords

Word embeddings; Text classification; Sentiment analysis

1. INTRODUCTION

Researchers have been making efforts for decades to represent text, from the simple count-based model, such as the bag-of-words (BOW) representation [9], also known as a one-hot representation, to statistical language models, such as the n-gram model [4]. However, limitations remain. The BOW model is known for lack of order information thus "Mary owes Bill $20" has an equivalent representation as "Bill owes Mary $20". Compared to the BOW model, the n-gram model retains the much of the spacial ordering, but n-grams still lack the ability to encode the semantic information in the vector representation. With recent advances in computer hardware, especially in GPUs, researchers are now able to build complex models on large datasets. Many approaches in the literature continue this line of research by trying to encode more semantic information into a distributed vector representation of words. Neural network language models (NNLM) [2] produce a distributed vector representation of word, also known as a word embedding. Compared to n-gram model, the neural language model does not rely on the count of the n-grams in the dataset but utilizing the neural network model to predict the word from the words appearing ahead of it [2], thus words with similar context will be mapped to close vector spaces. An efficient extension is the work from Mikolov et al. [15], also known as Word2Vec. Word2Vec consists of two model architectures, "skip-gram" and "continuous bag-of-words" (CBOW). Its simplified structure and reduced computational complexity make it widely popular. Word2Vec has been proved to be effective in many natural language tasks, such as name entity recognition (NER) [22] and dependency parsing [13]. Many researchers continue this line of investigation to build low-dimensional vector representation [20, 23]. These distributed vector representations of words serve as effective features in many text classification tasks.

1. (a) This is a chair with very **good** quality.
   (b) This is a chair with very **bad** quality
2. (a) I decided to buy the **apple** without considering the others.
   (b) This is the **case**.
   (c) The **water** level is rising.

By utilizing context information, word embeddings learned by neural language models are effective enough to

represent the general meaning of the words. However, the most serious problem is that by only considering the context of the word, words found in a similar context are placed similarly and words with multiple senses are overwhelmed by the words' dominant meaning. For example, the context of "good" and "bad" in Examples 1a and 1b are the same; thus words even with opposite polarities can be mapped to close vector space. In Example 2a, "apple" can be interpreted either as fruit or as computer brand; same with "case" in Example 2b; "water" in Example 2c can be interpreted as flood or water in the sink[1]. But in neural embeddings, only the dominant meaning will be represented. Thus when it comes to a general classification task, such as sentiment analysis, we would like to train word embeddings that separate enough in the vector space even when given identical contexts. General word embeddings are not effective enough for scenarios like Examples 1a and 1b. To tackle this problem, researchers train sentiment-specific embeddings [23] in a supervised approach, so that the embeddings are encoded with semantic information from the class label. A more serious problem is with polysemy in task-specific classification. Take hurricane-Sandy related tweet classification for example. In Example 2c, "water" should be interpreted as flood instead of rain or drinking water. But for other tasks, "water" might be interpreted as fluid, chemical liquid or the general meaning of water. Many research deals with ambiguity of words according to word's common senses from outside knowledge base [25, 5] or a small context window around the word [17, 21]. We found words' senses are hard to figure out given a short context (a sentence). We found that the meaning of a word in a sentence can be better interpreted with the help of class information or the label of the sentence.

To tackle the problems we mentioned above, we modify the skip-gram model in Word2Vec and build our own neural language model to train a class-specific word embedding. Our contributions include:

1. Our work is the first to use the linear composition property to build class-specific embeddings.
2. We present a general framework consisting of two convolutional neural networks which take the class-specific word embeddings we trained as input for a binary text classification task.

We compare our approach with multiple baselines on a disaster-related Twitter dataset and a benchmark Twitter sentiment classification dataset from SemEval 2013.

2. RELATED WORK

There are many methods to obtain vector representation of words, such as Latent Semantic Analysis (LSA) [12] and Latent Dirichlet Allocation (LDA) [3]. Word embedding trained by neural language models are well-known for its fine-granularity to represent words' general semantic meaning. More recently, Word2Vec, developed by Mikolov et al. [15], has shown to provide a new state-of-the-art performance in NLP tasks. Many researchers have contributed to the area of neural language model based word embedding [7, 23, 14, 6].

One problem the researchers found is that the current neural language based models are based on one assumption that a word's semantic meaning can be learned from the context in which it resides. However, as we show in Examples 1a and 1b, words with contrasting polarity but sharing similar contexts are mapped to close vector space positions. Tang et al. [23] tackle this problem by encoding the class information through modifying Collobert's C&W [7] model to a supervised approach.

[1]Example 2c is extracted from the disaster-focused Twitter corpus T6 [19].

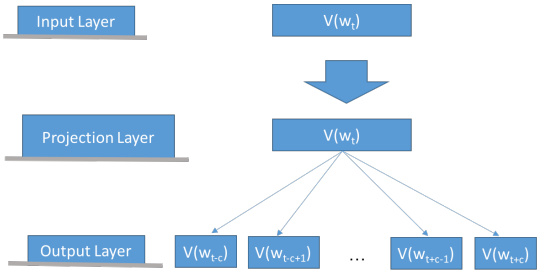

Figure 1: Skip-gram model architecture (from Mikolov et al. [15]).

Another problem is words with multiple meanings (polysemy). A single word embedding is insufficient to address the polysemy problem. Researchers address this issue by training multiple word embeddings per word according to their multiple senses [24, 18]. Hang et al. [10] tackle this problem by incorporating both local and global document context. Huang et al. used an outside knowledge base, WordNet [8] to obtain different senses of the words. However, we think that a word's sense such as "water" can be better interpreted with the aid of class information. For example, when the tweet "The water level is rising" is labeled as "hurricane-related tweet", we would know that the "water" here means flood.

In this work we propose to address these two problems by utilizing the linear compositionality property. We propose three approaches. In the first approach, we build separate word embeddings using data filtered by class label and feed the embeddings into the classification framework for a single class label prediction. In the second approach, we build class-specific word embedding by directly adding the vector representation of the classification polarity to the vector representing the general meaning of the word. In the third approach, we modify the skip-gram architecture to train a class-specific word embedding.

3. CLASS-SPECIFIC WORD EMBEDDINGS FOR TEXT CLASSIFICATION

In this section, we introduce the details of generating class-specific word embeddings. We propose to incorporate class information into word embeddings by utilizing the linear compositionality property shown by the word embeddings learned from neural network based language models [15, 20]. Our work directly extends the Word2Vec model architecture [15]. In the following sections, we present three model architectures to generate class-specific word embeddings. We then describe the use of the class-specific word embeddings in a framework consisting of convolutional neural networks for text classification.

3.1 Skip-gram Model

Mikolov et al. [15] introduce the skip-gram model, which learns the continuous vector representation of words from

the context in which the words reside. Figure 1 shows the skip-gram model's architecture. The model takes word w_t as input and predicts the c words ahead of and behind w_t by maximizing the log likelihood function:

$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log\, p(Context(w) \mid w)) \tag{1}$$

Where $\mathcal{C}$ is the corpus, the function tries to maximize the conditional probability of words appearing within a certain range of w given w. To address computational complexity, Mikolov et al. adopts hierarchical softmax and negative sampling [15] to implement skip-gram model. In this paper, we focus on skip-gram model trained with hierarchical softmax. The vocabulary in the skip-gram model with hierarchical softmax is initialized as a Huffman tree.

While Mikolov et al. present both the continuous bag of words (CBOW) and skip-gram models, we have focused on the skip-gram model. One reason is that the skip-gram model generally performs better in semantic tests [15] in terms of accuracy though slower compared to CBOW. The other reason is that compared to CBOW, skip-gram model trains over more data since each word in the corpus can be a training tuple. Thus the skip-gram model favors small datasets. In our work, we use a labeled dataset to encode the class information into embeddings. Labeled datasets are usually smaller (because of the cost to acquire the labels), and thus the skip-gram model is the appropriate choice.

3.2 Class-Specific Word Embedding

Neural network language models assume words that appear in a similar context are also semantically close to each other [15]. But with this assumption, words with (almost) syntactic equivalence and contrasting semantic polarity can be mapped closely in vector space. As we described earlier, this approach is also problematic for classification tasks associated with polysemy. In this section, we describe our proposed model based on the linear compositionality property of modern word embeddings.

3.2.1 Linear compositionality property

Word embeddings learned from the skip-gram model show good linear compositionality [15, 16]. A famous example would be that

$$vector(\text{“}King\text{”}) - vector(\text{“}Man\text{”}) + vector(\text{“}Woman''\text{)}$$

results in a vector which is the closest to the vector representation of the word “Queen” [15]. One interpretation is that the operation of $vector(\text{“}King''\text{)} - vector(\text{“}Man''\text{)}$ results in a vector which is the closest to the semantic definition of “Royalty”; thus $vector(\text{“}King''\text{)} - vector(\text{“}Man''\text{)} = vector(\text{“}Royalty''\text{)}$; then we add $vector(\text{“}Royalty''\text{)}$ to $vector(\text{“}Woman''\text{)}$, we get the semantic information from both words, which is $vector(\text{“}Queen''\text{)}$. Based on this observation, the semantic information in the vector representation trained from a neural language model satisfies linear composition. Thus the vector representation of a word, such as “Queen”, which combines the semantic meaning of “Woman” and “Royalty”, could be obtained directly through addition operation. We observed that the new vector representation combines the semantic definitions of both sides. In our work, we use this linear compositionality property to encode the class information into the word embeddings by adding the vector that represents the class information.

3.2.2 Vector representation of class

Based on the linear composition property, we propose to obtain the class-specific word embedding by adding the vector representation of the class to the vector that represents the general meaning of the word. In this section, we describe how we find the vector representation of the class information.

In the procedure to compute the vector representation of class, our first step is to manually define the classification polarity of the task. Some classification tasks have one polarity while others have two or more polarities. For example, in a basic sentiment analysis task, there are typically two polarities in this task, namely positive and negative; in a task to classify hurricane related Tweets from general Tweets stream, there is only one polarity, namely hurricane. Because for tweets that are labeled as hurricane-unrelated, we treat them as ordinary tweets which have no semantic polarity inside the sentences in terms of this task. In the second step, we manually find the word that is most representative of classification polarity of the task. We define the word as polarity word, such as “hurricane”.

In the third step, we adopt an heuristic approach to find the vector representations of the classification polarities. We first use the original skip-gram model from Word2Vec on our dataset to obtain class-independent word embeddings, providing a vector representation for each word in our vocabulary on the dataset. From the class-independent word embeddings we retrieve the polarity words' vector representations. Next, for each polarity word we use cosine similarity to select the top n words' vector representations that are most similar to the polarity word's vector representation from the vocabulary of the dataset:

$$\text{similarity\_score} = \frac{\text{vector}(w_{polarity}) \cdot \text{vector}(w)}{\|\text{vector}(w_{polarity})\| \|\text{vector}(w)\|} \tag{2}$$

where w is a word in the vocabulary; $w_{polarity}$ is the polarity word. According to the similarity score, we choose n vector(w) which have highest similarity scores. Then we calculate the arithmetic mean of the top n vector(w) as the vector representation of the class:

$$\mathcal{V}(class) = \frac{1}{n}(\text{vector}(w_1) + \text{vector}(w_2) + \cdots \text{vector}(w_n)) \tag{3}$$

where $vector(\cdot)$ denotes the embedding's vector representation of a word; $\mathcal{V}(\cdot)$ denotes the vector representation of class information. In our work n is 100.

3.2.3 Basic Approach I

In basic approach I, we do not use the vector representation of class to build class-specific word embedding. Our idea is simple: we first divide the training set into subsets according to the class label; for example, in a sentiment analysis dataset, the training set is divided into 2 subsets according to class label, namely positive set and negative set; next we train a skip-gram model over the data in each subset to generate a particular set of word embeddings for a specific class; in the sentiment analysis example, we generate one set of word embedding on the positive set and we generate another set of word embedding on the negative set using skip-gram model. So for each class, we have a separate set of word embeddings. We then apply the two sets of word embeddings to our parallel CNN classification framework.

This approach is designed to allow us to test the effective-

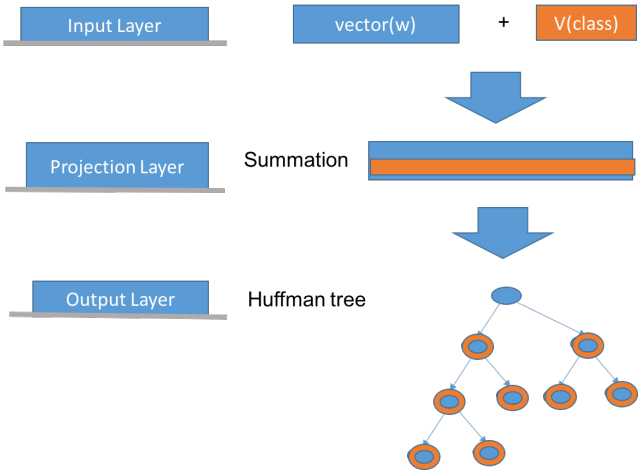

Figure 2: Advanced model architecture

ness of the linear compositionality property. We expect that on the same dataset training the embedding without utilizing the linear compositionality property would dampen the classification framework's performance.

3.2.4 Basic Approach II

As an unsupervised approach, the skip-gram model does not utilize the class information to learn word embeddings. For a binary classification task, we aim to train two vector representations for each word; one for each class. In basic approach II, we use linear composition to encode the class information into general word embeddings.

In basic approach II, we integrate the class information into the general word embedding by directly adding the vector representation of the classification polarities to the vector representing the general meaning of the word based on the linear compositionality property. For example, in a task to classify hurricane related tweets from a general tweet stream, we have elements of the training dataset labeled as "hurricane-related" or "hurricane-unrelated". Since there is only one polarity word "hurricane", we obtain the vector representation of the class hurricane according to Section 3.2.2; then the class-specific word embedding is defined as:

$$vector_c(w) = \mathcal{V}(hurricane) + vector(w) \tag{4}$$

where $vector_c(w)$ denotes the class-specific word embedding of word w; $vector(w)$ denotes the vector representing the general meaning of word w trained from skip-gram model.

3.2.5 Advanced Model

Based on the observation of linear compositionality property of the word embeddings trained on neural language model, our basic approach II generates a class-specific word embedding by adding directly $\mathcal{V}(\cdot)$, the vector representation of class information to $vector(w)$, the vector representation of the general meaning of word w. Our goal is to combine the class information into word embedding to incorporate a global context information that would otherwise be missed in short texts such as tweets. However, although the basic approach II tackles the polysemy problem by adding the class information, the problem brought up by words with contrasting polarity but similar context remains unsolved. The reason is that in the vector space, we move all the words with the same distance by $\mathcal{V}(\cdot)$. If words with contrasting polarities were previously close in vector space, they are still close to each other under the basic approach II. To tackle this problem, in the advanced model we adapt the model architecture from skip-gram to train class-specific word embedding. Instead of adding the class information vector linearly, we utilize the neural network model to predict the context words' embeddings.

Figure 2 shows the architecture of the advanced model. Compared to basic approach II, we utilize the skip-gram model's neural network architecture to predict the context words' class-specific embeddings. Using the approach introduced in Section 3.2.2 to represent the class information in vector space denoted as $\mathcal{V}(class)$, we add $\mathcal{V}(class)$ to the general vector representation of the input word, $vector(w)$. Based on linear compositionality property, the summation of $\mathcal{V}(class)$ and $vector(w)$ should capture the semantic meaning from both sides. Thus armed with the class information from the label of the training tuple, we are able to predict more precisely the context words around the input word. The objective function for the advanced model is:

$$\mathcal{L} = \sum_{w \in \mathbb{C}} \log\, p(Context(w) \mid vector(w) + \mathcal{V}(class)) \tag{5}$$

where over all training tuples in the corpus $\mathbb{C}$, we are maximizing the probability of finding the context words around w given w and its class information. We adopt hierarchical softmax based skip-gram model [15], which uses a binary Huffman tree to organize the words in the vocabulary. Each leaf in the Huffman tree represents a word. The path from root to leave represents the Huffman encoding of the word. And for each non-leaf node in the tree, it is a binary classification that produce a probability to decide which path to take. As Mikolov et al's skip-gram model, we choose logistic regression classifier for each non-leaf node. Thus the conditional probability in Equation 5 can be further written down as:

$$p(Context(w) \mid w, \mathcal{V}(class)) = \prod_{j=2}^{l^w} p(d_j^w | X_{w,\mathcal{V}(class)}, \theta_{j-1}^w) \tag{6}$$

$$p(d_j^w | X_{w,\mathcal{V}(class)}, \theta_{j-1}^w) = \begin{cases} \sigma(X_{w,\mathcal{V}(class)}), & d_j^w = 0 \\ 1 - \sigma(X_{w,\mathcal{V}(class)}), & d_j^w = 1 \end{cases} \tag{7}$$

where $X_{w,\mathcal{V}(class)}$ denotes the output of the projection layer in the advanced model, which is the summation of $vector(w)$ and $\mathcal{V}(class)$; d_j^w is the binary Huffman code at jth node of word w; θ_{j-1}^w is the vector representation of the $(j-1)th$ non-leaf node of word w; l^w is the number of non-leaf nodes for word w; $\sigma(\cdot)$ is the sigmoid activation function of the logistic regression classifier at non-leaf nodes. We use SGD (Stochastic Gradient Descent) to maximize $\mathcal{L}$ and update $X_{w,\mathcal{V}(class)}$ and θ_{j-1}^w.

3.3 Classification Framework

We apply class-specific word embedding for text classification under a supervised learning framework. Our framework extends Kim's work [11]. Kim introduced the use of convolutional neural networks (CNN) for sentence classification. In Kim's work, the input is a sentence. For each word in the sentence, Kim's CNN takes one fixed length of word embed-

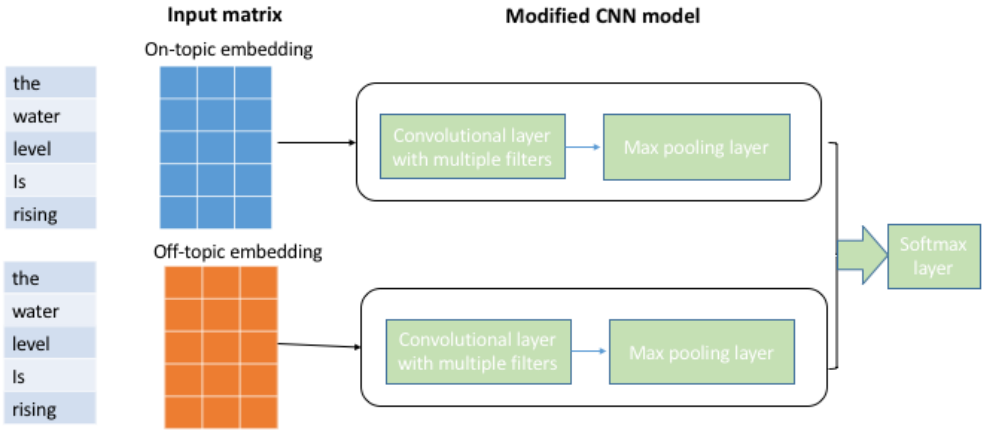

Figure 3: A general binary classification framework that takes two embeddings, namely on-topic embedding and off-topic embedding as input

ding trained from Word2Vec. It consists of a convolutional layer with multiple filters in different width, a max-pooling layer and a softmax output layer.

We aim to build a classifier framework which takes the class-specific word embedding we trained as input. Since for each test sentence its class label is not revealed yet, it is not certain that which word embedding, for example class-specific word embedding or general meaning word embedding, should be applied to a classifier. We design a classification framework that takes multiple sets of word embeddings as input. The number of word embedding per word depends on the class polarities of the classification task. For example, for sentiment analysis we have two class polarities, thus we have two word embeddings per word: one embedding learned from positive class and the other embedding learned from negative class. For a topic-related classification task, such as a task to classify hurricane-related tweets, we also have two word embeddings per word: one on-topic embedding trained from hurricane-related tweets and one off-topic embedding trained from hurricane-unrelated tweets.

Take binary text classification for example, the proposed classification framework is illustrated in Figure 3. We combine two CNNs with a softmax layer which takes concatenated feature vectors from the two max pooling layer and outputs the probability distribution over class labels.

4. EXPERIMENT

We conduct experiments to evaluate the proposed models to learn class-specific word embeddings. We apply class-specific word embeddings to the supervised classification framework described in Section 3.3.

4.1 Experiment Setup and Datasets

We conduct experiments on two publicly available datasets. The first dataset is a disaster-related Twitter dataset [19], called T6. T6 is labeled by crowdsourcing workers according to disaster relatedness (as "on-topic", or "off-topic") [19]. T6 contains 6 crisis events in 2012 and 2013. We choose to test our approach on the hurricane Sandy dataset. The statistics of the hurricane Sandy dataset is shown in Table 1. The other dataset is the benchmark Twitter sentiment classification dataset in SemEval 2013[2]. For each tuple in the SemEval dataset, it has three class label options, namely, positive, negative and neutral. Since we focus on the binary text classification task and we aim to use the same classification framework for both of the datasets, we filter out the tuples in the SemEval 2013 dataset which are labeled as neutral. We also do a pre-processing step: we first eliminate all the tweets in the two datasets that are non-English, and then we eliminate tweets that contain fewer than five words. Our pre-processing step is in line with Olteanu et al's work on the same dataset [19]. For the parameters of our experiments, we choose a window size of 5 and word embedding dimension of 50. The characteristics of the SemEval 2013 dataset are shown in Table 2.

Data	On-topic	Off-topic	Total
Train	4911	3098	8009
Test	1227	772	1999

Table 1: Hurricane Sandy dataset characteristics

Data	Positive	Negative	Total
Train	2256	849	3104
Test	330	172	502

Table 2: SemEval 2013 dataset characteristics

4.2 Baseline Methods

To compare the quality of the class-specific word embedding, we choose the following baselines:

1. Sentiment-specific word embedding (SSWE): Tang et al. [23] introduce a supervised method to learn sentiment-specific word embedding based on Collobert et al's unsupervised approach [7]. We build a word embedding according to Tang's method and test the embedding on our classification framework. We use Attardi's NLP pipeline to generate this baseline [1].

[2]https://www.cs.york.ac.uk/semeval-2013/

Method	Hurricane Sandy	SemEval 2013
SSWE+CNN [23]	85.54	69.92
Skip-gram [15] created embedding using unlabeled text + single CNN [11]	86.00	70.72
Basic Approach I: Two skip-gram-generated embeddings from class-filtered text + parallel CNN framework	86.94	72.51
Basic Approach II: Addition of class vector and general meaning vector + parallel CNN framework	86.95	72.11
Advanced Model: Modified skip-gram + parallel CNN framework	86.99	73.70

Table 3: Comparison of classification accuracy across the two datasets using word embeddings from various models

2. Word embedding trained using the skip-gram model: we train our own embedding using Word2Vec's original skip-gram model [15]. We apply the word embedding as features of a convolutional neural network [11]. A single embedding per word is trained on all training data without use of the training labels.

Tang et al.'s approach [23] is the research work that is closest to our own. Although their method is to generate a sentiment-specific embedding, we found their method could be extended to any labeled dataset that has two contrasting polarities.

4.3 Results and Analysis

Table 3 shows the results of the experiments on different approaches. We choose convolutional neural networks over other classifiers. The reasons are: since our proposed classification framework consists of two CNNs, thus it is reasonable to compare our framework's performance with a single CNN; secondly, a CNN has achieved the state-of-art result in sentiment analysis as shown by Kim [11].

For the SSWE baseline, we use Attardi's implementation [1] of SSWE [23] to generate hurricane-specific and sentiment-specific word embeddings. Although our work and SSWE are both derived from neural language models, our model extends Mikolov's skip-gram model, while SSWE extends Collobert's C&W model [7]. The skip-gram model has a simple architecture, while C&W model keeps a look-up table for all the words in the vocabulary and a fully-connected hidden layer, which makes SSWE slower to compute and hard to scale to large datasets. In Basic Approach II, we use the tweets in the training set to generate a general meaning word embedding $vector(w)$. We then calculate $\mathcal{V}(hurricane)$ and add $\mathcal{V}(hurricane)$ to $vector(w)$ to produce a class-specific embedding for the second CNN. We then use the two sets of embeddings in the classification framework. In the advanced model, we use the same $\mathcal{V}(hurricane)$ from Basic Approach II and added to the input word for each training tuple in the input layer to train the class-specific word embedding.

In the SemEval dataset experiments, a slight difference is in the choice of the polarity word when we try to calculate $\mathcal{V}(positive)$ and $\mathcal{V}(negative)$. We choose the polarity word "good" for positive class and "bad" for negative class for use in generating two sets of class-specific word embeddings for Basic Approach II and the Advanced Model.

In both sets of experiments, the SSWE+CNN result is relatively weak compared to skip-gram based models. The result of Basic Approach I using two sets of self-trained embedding on our framework is better than the result of the second baseline, which uses one single CNN. We ascribe the reason to be that we combine more classifiers that use different features (e.g., from different embeddings). It is similar to the ensemble method in machine learning, thus improve the overall performance. The result of the Basic Approach II is very similar to the result of the Basic Approach I. This indicates part of our concern that only shifting all the embedding in the vector space by the same distance is hard to boost the performance. Results for the Advanced Model is the highest, outperforming both baselines and the basic approaches.

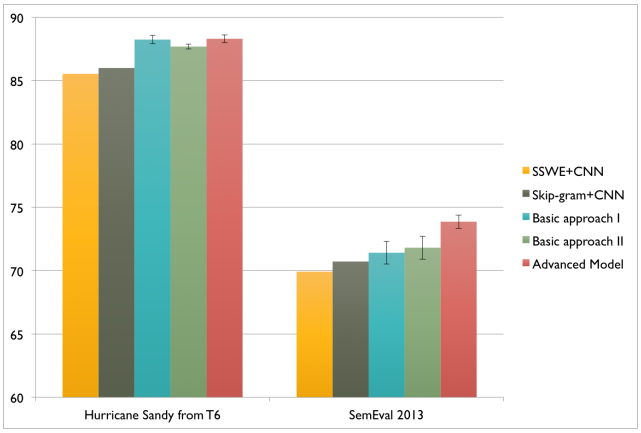

Figure 4: Mean classification accuracy of ten additional runs for the proposed models, with standard deviations, compared to the two existing approaches.

There is stochasticity in the three proposed approaches. For example, all of the embeddings are initialized with random values. To reduce the uncertainty in our measured results, we run every proposed model an additional ten times to obtain an expected classification accuracy. The results are shown in Figure 4. For the Hurricane Sandy dataset, the mean Basic Approach I accuracy was more than 2 percentage points higher than the results in Table 3. For the SemEval 2013 dataset, although still higher than the baselines, the mean result decreases by approximately one percent. For both datasets, the advanced model remains the best performer, achieving a mean relatively improvement of 3.2% (Hurricane Sandy) to 5.6% (SemEval 2013) over the SSWE+CNN baseline. This suggests our proposed approaches to generate class-specific word embedding combined with the parallel CNN framework can improve the performance on text classification tasks.

4.4 Discussion

To choose "good" and "bad" as the polarity words is risky. We found in the Twitter dataset, people describe positive and negative emotion using lexicons with great variety, such as "Gas by my house hit $3.99!! I'am going to Chapel Hill on Sat!", "Twitition Mcfly come back to Argentina but this time we want to come to mar del plata!!!" and "Never start working on your dreams and goals tomorrow.....tomorrow never comes....if it means anything to U, ACT NOW! #getafterit"[3] These three tweets have no lexicon that are associated with "good" or "bad". Thus how to choose or generate polarity words to produce vector representation of the class is still an open question.

To calculate the vector representation of class information, we average the embeddings of the top 100 words. The number 100 is a hyper-parameter, and how to choose it optimally remains an open issue.

Another observation is that summation is best suited for elementary words such as "water". When an elementary word is added to a complex-meaning word, such as "massacre", we found the meaning of the elementary word is often overwhelmed by the complex-meaning word. This problem is best demonstrated by finding the most n similar words from the vocabulary using cosine similarity. When we add $vector(water)$ to $vector(massacre)$, the top ranked words are "massacres", "killings" and "murders". The semantic of word "water" seems to have disappeared, which introduces another research problem.

5. CONCLUSION

In this paper, we propose to use the linear compositionality property to learn class-specific word embeddings for a text classification task. We devise a classification framework to take multiple sets of word embeddings as input. We test our methods on two Twitter datasets, namely a disaster-related dataset and SemEval 2013. Our results show that for text classification tasks with clear polarity words, our proposed approaches can increase performance.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-1541177.

6. REFERENCES

[1] G. Attardi. DeepNL: a deep learning NLP pipeline. In *Proceedings of NAACL-HLT*, pages 109–115, 2015.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[4] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[5] X. Chen, Z. Liu, and M. Sun. A unified model for word sense representation and disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, Doha, Qatar, October 2014. Association for Computational Linguistics.

[6] Y. Chen, B. Perozzi, R. Al-Rfou, and S. Skiena. The expressive power of word embeddings. In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[8] C. Fellbaum. *WordNet*. Wiley Online Library, 1998.

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

[10] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

[11] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1746âĂŞ1751, Oct. 2014. arXiv preprint arXiv:1408.5882.

[12] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284, 1998.

[13] O. Levy and Y. Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308, 2014.

[14] W. Ling, C. Dyer, A. Black, and I. Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, 2015.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[16] J. Mitchell and M. Lapata. Vector-based models of semantic composition. In *Proceeding of the Annual Meeting of the Association for Computational Linguistics*, pages 236–244, 2008.

[17] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[18] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*, 2015.

[19] A. Olteanu, C. Castillo, F. Diaz, and S. Vieweg. Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, June 2014.

[20] J. Pennington, R. Socher, and C. D. Manning. GloVe:

[3]These three tweets are extracted from SemEval 2013 training data.

Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[21] J. Reisinger and R. J. Mooney. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics, 2010.

[22] S. K. Sienčnik. Adapting word2vec to named entity recognition. In *Proceedings of the 20th Nordic Conference of Computational Linguistics, Vilnius, Lithuania*, number 109, pages 239–243. Linköping University Electronic Press, May 2015.

[23] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for Twitter sentiment classification. In *Proceeding of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[24] A. Trask, P. Michalak, and J. Liu. sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. *arXiv preprint arXiv:1511.06388*, 2015.

[25] M. Yu, M. Gormley, and M. Dredze. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*, 2014.