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." Most word representation learning models are based on this assumption that a word's semantic meaning can be learned from the context in which it resides. The context is defined as a small unordered number of words surrounding the target word. Research has shown that context alone provides limited information because the context contains only neighboring words. Thus only local information is learned in the word embeddings. Some research tries to improve this by utilizing outside information sources such as a knowledge base. We observe that the meaning of a word in a sentence can be better interpreted when the class information or label of the sentence is presented. 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 where the learned class-specific embeddings serve as features. 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.**

***Index Terms*—Word embeddings; Text classification; Sentiment analysis**

I. Introduction

Word vector representations learned from neural language models, also known as word embeddings, are well-known for representing words' fine-grained semantic meaning [1], [2]. Word embeddings have been demonstrated to be useful in many machine learning and natural language processing tasks, such as part-of-speech tagging and sentiment analysis [3], [4], [5]. An efficient state-of-art implementation is the work from Mikolov et al. [2], also known as Word2Vec. Word2Vec consists of two neural language model architectures: "skip-gram" and "continuous bag-of-words" (CBOW). Its simplified structure and reduced computational complexity made it widely popular. Word2Vec has been proven to be effective in many natural language tasks, such as name entity recognition (NER) [6] and dependency parsing [7]. Many researchers continue this line of investigation to build other word vector representation learning models [4], [8]. These distributed vector representations of words serve as effective features in many text classification tasks [9].

Most word representation learning models are based on the assumption that a word's semantic meaning can be learned from the context in which it resides. The context is defined as a small unordered number of words surrounding the target word. Research has shown that context alone provides limited information because the context contains only neighboring words. Thus only local information is learned in the word embeddings [10], [11]. Some research tries to improve this by utilizing outside information sources such as a knowledge base (WordNet, PPDB, etc.) [11], [12], [13]. The outside knowledge-base-enhanced word embedding shows improvements in many natural language processing tasks, such as word similarity measurement, name entity recognition and dependency parsing [14], [15], [16]. However, other types of knowledge sources can also be explored in the word embedding learning process. In this paper we explore how class label information can help to enhance word embedding semantically.

1) I decided to buy the **apple** without considering the others.
2) This is the **case**.
3) The **water** level is rising.

Another problem is that most of the word representation learning models only learn one vector representation per word. This is problematic because many words are polysemous. In Example 1, "apple" can be interpreted either as fruit or as computer brand; same with "case" in Example 2; "water" in Example 3 can be interpreted as flood or water in the sink.[1]But in the unsupervised word vector representation learning, only one general meaning is learned based on the majority of the contexts of the target word appearing in the corpus. General word embeddings are not effective enough for scenarios like Examples 1, 2 and 3. To tackle this problem, researchers train sentiment-specific embeddings [8] 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 3, "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. Much research attempts to deal with ambiguity of words according to word's common senses from outside knowledge bases [18], [19] or a small context window around the word [20], [21]. We found words' senses are hard to determine given a short context (a sentence). We observed 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. For example, when the tweet "The

[1]Example 3 is extracted from the disaster-focused Twitter corpus T6 [17].

water level is rising" is labeled as "hurricane-related tweet", we would know that the "water" here likely means a flood. Thus instead of training one vector representation per word, we train the number of word vector representations according to how many classes (labels) the corpus has. Both corpora in our experiments are used in binary classification. In our work we train two vector representations per word.

We modify the skip-gram model in Word2Vec and build our own neural language model to train a class-specific word embedding. We utilize the linear composition property of the word embedding to encode class information. 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 evaluate our approach and framework on a disaster-related Twitter dataset and a benchmark Twitter sentiment classification dataset from SemEval 2013. 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.

II. Related Work

There are many methods to obtain a vector representation of words, such as Latent Semantic Analysis (LSA) [22] and Latent Dirichlet Allocation (LDA) [23]. 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. [2], 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 [24], [8], [25].

The current neural language based models are based on the assumption that a word's semantic meaning can be learned from the context in which it resides. This assumption generally holds. However, many words are polysemous. Local context alone can hardly figure out which sense of the word is used in the sentence. Tang et al. [8] tackle this problem by encoding the class information through modifying Collobert's C&W [24] model to a supervised approach.

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 [26], [27]. Hang et al. [10] tackle this problem by incorporating both local and global document context. Huang et al. used an outside knowledge base, WordNet [28] to obtain different senses of the words. To let the model learn automatically the number of vector representation that a polysemous word should have, Zheng et al. developed an algorithm to learn a new sense vector for a word if the cosine similarity between the new emerged context vector and every existing sense vector is less than a threshold [29]. Tian et al. extended the skip-gram model from Mikolov's work and generated multiple vector representations for each word in a probabilistic manner [30]. 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.

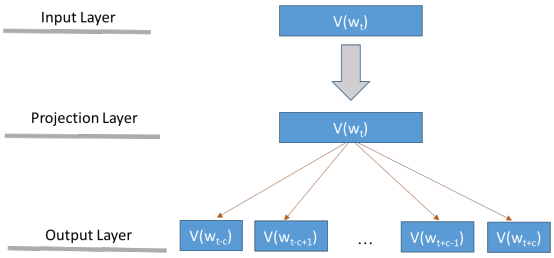

Fig. 1. Skip-gram model architecture (from Mikolov et al. [2]).

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.

III. 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 [2], [4]. Our work directly extends the Word2Vec model architecture [2]. 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.

A. Skip-gram Model

Mikolov et al. [2] 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:

$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log\, p(w_{t-m}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+m} \mid w_t)) \quad (1)$$

Where $\mathcal{C}$ is the corpus, the function tries to maximize the sum of the conditional probability of m words preceding and succeeding w_t. To address computational complexity, Mikolov et al. adopts hierarchical softmax and negative sampling [2] 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.

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 [2] 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.

B. Class-Specific Word Embedding

Neural network language models assume words that appear in a similar context are also semantically close to each other [2]. As we described earlier, this assumption can be problematic because local context alone can only provide limited information because the context contains only neighboring words in a fixed-size window around the target word. In this section, we describe our proposed model based on the linear compositionality property of modern word embeddings.

1) Linear compositionality property: Word embeddings learned from the skip-gram model show good linear compositionality [2], [31]. 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” [2]. 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.

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, namely positive and negative; in a task to classify hurricane-related tweets from a general tweet 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 a 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:

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

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:

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

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.

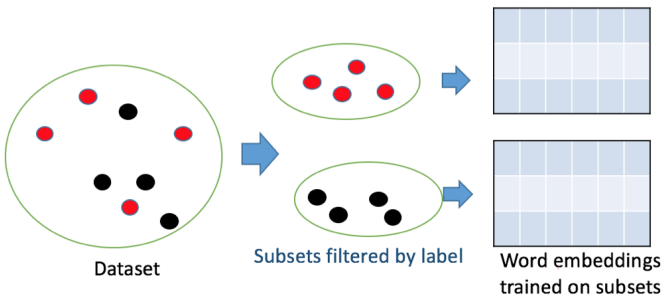

Fig. 2. Diagram of Basic Approach I.

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 as shown in figure 2: 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 effectiveness 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.

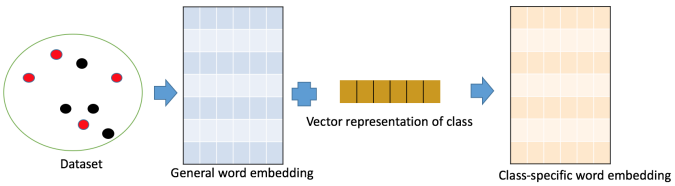

Fig. 3. Diagram of Basic Approach II.

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 as illustrated in figure 3.

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 III-B2; 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.

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

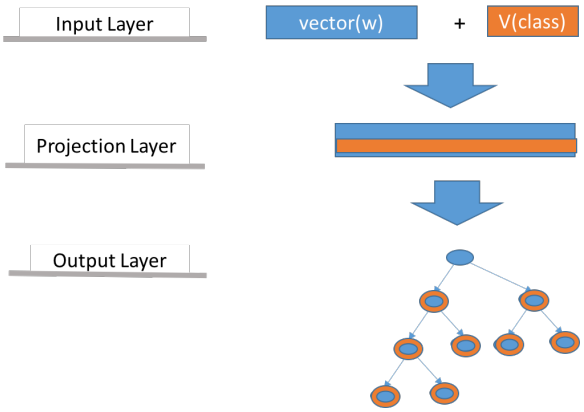

Fig. 4. Advanced model architecture.

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. In the advanced model we extend 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 4 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 III-B2 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 [2], 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.

C. Classification Framework

We apply class-specific word embedding for text classification under a supervised learning framework. Our framework extends Kim's work [32]. 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 embedding 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 5. 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.

IV. 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 III-C.

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

TABLE I
HURRICANE SANDY DATASET CHARACTERISTICS.

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

TABLE II
SEMEVAL 2013 DATASET CHARACTERISTICS.

A. Experiment Setup and Datasets

We conduct experiments on two publicly available datasets. The first is a disaster-related Twitter dataset [17], called T6. T6 is labeled by crowdsourcing workers according to disaster relatedness (as "on-topic", or "off-topic") [17]. T6 contains 6 crisis events in 2012 and 2013. We choose to test our approach on the hurricane Sandy dataset. The characteristics of the hurricane Sandy dataset are shown in Table I. 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 [17]. For the parameters of our experiments, we choose a window size of 5 and word embedding dimension of 50. To reduce the randomness and the stochasticity in the experiments, we conduct each experiment 30 times and report the mean results of the 30 runs for each experiment. The characteristics of the SemEval 2013 dataset are shown in Table II.

B. 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. [8] introduce a supervised method to learn sentiment-specific word embedding based on Collobert et al.'s unsupervised approach [24]. 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 [33].
2) Word embedding trained using the skip-gram model: we train our own embedding using Word2Vec's original skip-gram model [2]. We apply the word embedding as features of a convolutional neural network [32]. A single embedding per word is trained on all training data without use of the training labels.

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

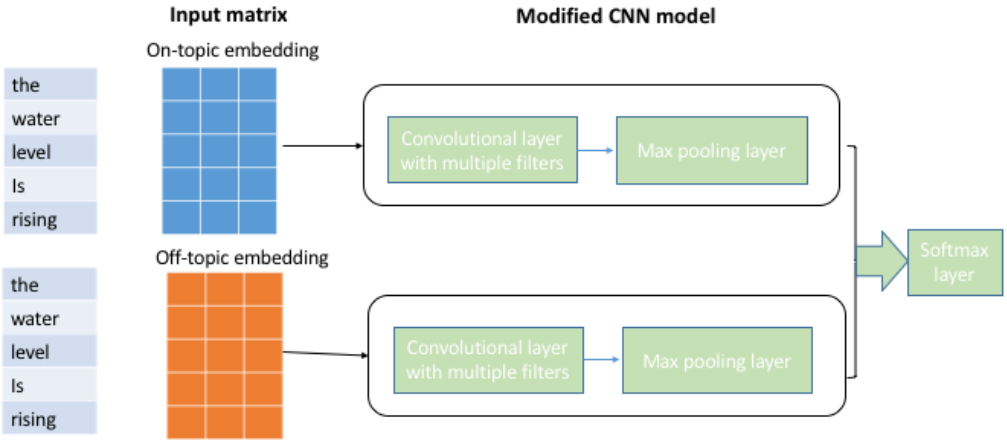

Fig. 5. A general binary classification framework that takes two embeddings, namely on-topic embedding and off-topic embedding as input.

Method	Hurricane Sandy	SemEval 2013
SSWE+CNN [8]	85.54	69.92
Skip-gram [2] created embedding using unlabeled text + single CNN [32]	86.00	70.72
Basic Approach I: Two skip-gram-generated embeddings from class-filtered text + parallel CNN framework	88.15	71.35
Basic Approach II: Addition of class vector and general meaning vector + parallel CNN framework	87.72	71.60
Advanced Model: Modified skip-gram + parallel CNN framework	88.19	73.15

TABLE III
COMPARISON OF CLASSIFICATION ACCURACY ACROSS THE TWO DATASETS USING WORD EMBEDDINGS FROM VARIOUS MODELS.

Tang et al.'s approach [8] 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.

C. Results and Analysis

Table III 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 [32].

For the SSWE baseline, we use Attardi's implementation [33] of SSWE [8] 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 [24]. 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.

There is stochasticity in the three proposed approaches. For example, all of the embeddings are initialized with random values. To better illustrate the thirty runs results of the proposed models, we also show the result in bar chart with standard

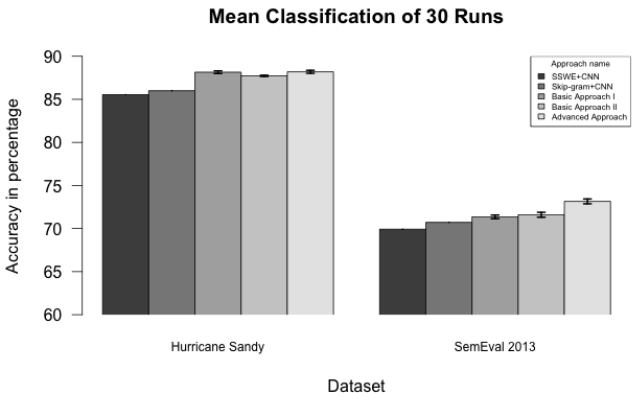

Fig. 6. Mean classification accuracy of thirty additional runs for the proposed models, with standard errors, compared to the two existing baseline approaches.

errors on the proposed models. The results are shown in Figure 6. We could see that compared to SemEval 2013 dataset, Hurricane Sandy dataset tends to have tighter error bars. For the Hurricane Sandy dataset, the mean Basic Approach I accuracy was more than 2 percentage points higher than the results in Table III. For both datasets, the advanced model remains the best performer, achieving a mean relative improvement of 3.1% (Hurricane Sandy) to 4.6% (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.

Moreover to better measure the performance of the class-specific word embedding we trained, we compare the word embedding trained from the advanced approach with the embeddings trained from skip-gram model, one of our baselines in the first dataset Hurricane Sandy. Most tuples in the test set that mention "hurricane","hurricane Sandy" and "Frankenstorm" are recognized correctly in both advanced approach and baseline models. For example, "Frankenstorm was actually the name of the creator. This hurricane should properly be called Frankenstorm's monster.", "Praying for everyone in the path of Hurricane Sandy." and "This hurricane blowing me now." To better verify the effectiveness of the proposed approach, we look at some tuples from test set that are classified correctly (True Positive) in advanced approach but classified incorrectly (False Negative) in the second baseline using skip-gram model as shown in 1, 2 and 3. We found that after adding class information in the advanced approach, tuples such as 1, 2 and 3 can be recognized as "hurricane-related" even without obvious words or hashtag indicators.

1) My power was out for like 5 days when Irene hit.
2) Stranger danger! My power's out too. Be safe #drinkingtillifallasleep
3) People on the other side of the country won't see specifically how #lbi is doing. It's all grouped into the east coast.

D. Parameter Sensitivity

In order to evaluate how changes to the parameterization of the proposed three approaches effect its performance on classification tasks, we conducted experiments on the two binary classification tasks (Hurricane Sandy dataset and SemEval 2013 dataset).

To only focus on the performance of parameterization, we need to remove the randomness introduced during the training procedure. We first fixed the seed parameter in the initializing of the word embedding vectors so that the experiments could be repeated with identical word vector initialization; then we used only a single thread to eliminate randomness introduced by operating system thread scheduling. To further reduce the randomness in word vector initialization, during the experiments of trials with different word vector dimensionality, we initialize the maximum dimension n of word vector so that each word vector with different dimensionality s will be initialized by selecting the top s numbers from the initialized word vector of size n. The result of these steps is a process that would repeatedly assign exactly the same random values regardless of the size of the word vector representation.

During all experiments, we fixed the window size to be 5. All experiments are performed starting from word vectors of dimension 5 to word vectors of dimension 165 with an interval of 10. Because of limited space, we are unable to plot the performance. For Hurricane Sandy dataset, the optimal dimensionality for Basic Approach I and the Advanced Approach is obtained near 65, while Basic Approach II is relatively steady throughout. The performance is quite consistent between both Hurricane Sandy and SemEval 2013 datasets in the sense that the Advanced Approach in both datasets shows a better performance at a lower dimensionality (around 65 for Hurricane Sandy and around 45 for SemEval 2013) compared to the other proposed approaches.

E. 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

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

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.

V. Conclusion

We explore how the class information or class labels can help to distinguish word senses. Enlightened by the good linearity property shown in the word vector representation learned from neural language models, we propose to use the linear compositionality property to learn class-specific word embeddings for a text classification task. We propose three approaches to learn class-specific word embeddings. 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.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[2] 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*, 2013, pp. 3111–3119.

[3] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks." 2014,Oct 25, pp. 740–750.

[4] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[5] Q. Liu, Z.-H. Ling, H. Jiang, and Y. Hu, "Part-of-speech relevance weights for learning word embeddings," *arXiv preprint arXiv:1603.07695*, 2016.

[6] S. K. Sienčnik, "Adapting word2vec to named entity recognition," in *Proceedings of the 20th Nordic Conference of Computational Linguistics, Vilnius, Lithuania*, no. 109. Linköping University Electronic Press, May 2015, pp. 239–243.

[7] O. Levy and Y. Goldberg, "Dependency-based word embeddings." in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, vol. 2, 2014, pp. 302–308.

[8] 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.

[9] S. Kuang and B. D. Davison, "Semantic and context-aware linguistic model for bias detection." in *Proc. of the Natural Language Processing meets Journalism IJCAI-16 Workshop*, July 10, 2016, pp. 57–62.

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

[11] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," *arXiv preprint arXiv:1411.4166*, 2014.

[12] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "From paraphrase database to compositional paraphrase model and back," *TACL*, vol. 3, pp. 345–358, 2015. [Online]. Available: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/571

[13] M. Yu and M. Dredze, "Improving lexical embeddings with semantic knowledge." in *ACL (2)*, 2014, pp. 545–550.

[14] Q. Liu, H. Jiang, S. Wei, Z.-H. Ling, and Y. Hu, "Learning semantic word embeddings based on ordinal knowledge constraints." in *ACL (1)*, 2015, pp. 1501–1511.

[15] D. Bollegala, M. Alsuhaibani, T. Maehara, and K.-i. Kawarabayashi, "Joint word representation learning using a corpus and a semantic lexicon." in *AAAI*, 2016, pp. 2690–2696.

[16] M. Bansal, K. Gimpel, and K. Livescu, "Tailoring continuous word representations for dependency parsing." in *ACL (2)*, 2014, pp. 809–815.

[17] 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*, Jun. 2014.

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

[19] 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)*. Doha, Qatar: Association for Computational Linguistics, October 2014, pp. 1025–1035. [Online]. Available: http://www.aclweb.org/anthology/D14-1110

[20] 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.

[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*. Association for Computational Linguistics, 2010, pp. 109–117.

[22] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse Processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[23] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[24] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[25] W. Ling, C. Dyer, A. Black, and I. Trancoso, "Two/too simple adaptations of word2vec for syntax problems," in *Proc. of the 2015 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1299–1304.

[26] 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.

[27] 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.

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

[29] X. Zheng, J. Feng, Y. Chen, H. Peng, and W. Zhang, "Learning context-specific word/character embeddings," 2017. [Online]. Available: https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14601

[30] F. Tian, H. Dai, J. Bian, B. Gao, R. Zhang, E. Chen, and T.-Y. Liu, "A probabilistic model for learning multi-prototype word embeddings." 2014, pp. 151–160.

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

[32] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Oct. 2014, p. 17461751, arXiv preprint arXiv:1408.5882.

[33] G. Attardi, "DeepNL: a deep learning NLP pipeline," in *Proceedings of NAACL-HLT*, 2015, pp. 109–115.