Novel Semantics-based Distributed Representations for Message Polarity Classification using Deep Convolutional Neural Networks

Abhinay Pandya and Mourad Oussalah
Center for Ubiquitous Computing, Faculty of Information Technology and Electrical Engineering (ITEE), University of Oulu, Finland

Keywords: Sentiment Analysis, Deep Learning, Information Retrieval, Text Mining.

Abstract: Unsupervised learning of distributed representations (word embeddings) obviates the need for task-specific feature engineering for various NLP applications. However, such representations learned from massive text datasets do not faithfully represent finer semantic information in the feature space required by specific applications. This is owing to the fact that (a) models learning such representations ignore the linguistic structure of the sentences, (b) they fail to capture *polysemous* usages of the words, and (c) they ignore pre-existing semantic information from manually-created ontologies. In this paper, we propose three semantics-based distributed representations of words and phrases as features for message polarity classification: Sentiment-Specific Multi-Word Expressions Embeddings(SSMWE) are sentiment encoded distributed representations of *multi-word expressions (MWEs)*; Sense-Disambiguated Word Embeddings(SDWE) are sense-specific distributed representations of words; and WordNet embeddings(WNE) are distributed representations of hypernym and hyponym of the correct sense of a given word. We examine the effects of these features incorporated in a convolutional neural network(CNN) model for evaluation on the SemEval benchmarked dataset. Our approach of using these novel features yields 14.24% improvement in the macro-averaged F1 score on SemEval datasets over existing methods. While we have shown promising results in twitter sentiment classification, we believe that the method is general enough to be applied to many NLP applications where finer semantic analysis is required.

1 INTRODUCTION

The use of microblogging and social network websites such as Facebook[1], Twitter[2], Tumblr[3] is prevalent in sharing diverse kinds of information which does not just include news and facts, but also expressing opinions and feelings. These platforms allow people to post real-time messages discussing a variety of topics. Twitter is certainly a leader among microblogging platforms: with over 1.3 billion users and about 500 million tweets posted per day and over 15 billion API calls per day, it provides a massive source for information analytics. One of the axes of such analytics is to gauge public opinion about a product, an event, a person, or an idea.

Message polarity classification is the task of determining whether the given textual message (e.g., a tweet) expresses a positive, a negative, or a neutral/objective sentiment with respect to a given contextual information. Applications of sentiment classification include but not limited to: understanding consumer perceptions (Smith et al., 2012), political opinion mining (Martinez-Camara et al., 2014), financial performance prediction (Bollen et al., 2011), and analyzing election outcomes (Skoric et al., 2012). (Mejova et al., 2015) discuss many socio-economic applications of Twitter sentiment analysis including public health, disaster management (Goodchild and Glennon, 2010), etc.

However, sentiment classification of tweets is difficult owing to the non-standard usage of language in tweets which are of a maximum length of 140 characters. In addition to having a poor grammatical structure, tweets contain slang words, misspellings, abbreviations, and hashtags which are not part of a standard vocabulary and thus pose a challenge to the automated analysis of tweets. Table 1 shows some examples to illustrate this.

[1]https://www.facebook.com
[2]https://www.twitter.com
[3]https://www.tumblr.com

Pandya A. and Oussalah M.
Novel Semantics-based Distributed Representations for Message Polarity Classification using Deep Convolutional Neural Networks.
DOI: 10.5220/0006500800710082
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KDIR 2017)*, pages 71-82
ISBN: 978-989-758-271-4

Table 1: Examples of tweets and their polarity from SemEval datasets.

@nater0driguez Lmfao alright u got me there. Good job Parker and the spurs, see y'all jan 9th. If I get an extra ticket to that game ur goin	**positive**
Realized that I've just spent Halloween, super bowl Sunday, and my last two birthdays either in the library, or in a computer lab..	**negative**
@NeilHarmanTimes Just noticed you using the Spanish abbreviation "NO sure", no? Think you're missing Rafa, mon brave	**negative**

1.1 Contributions

Various machine learning approaches have been developed over the past decade for twitter sentiment classification. Especially, since the introduction of twitter sentiment analysis (TSA) as a shared task in SemEval(Rosenthal et al., 2014; Rosenthal et al., 2015; Nakov et al., 2016), many new manually-labeled resources have become available and the competitive setup of such tasks have given rise to a number of new approaches. Lately, after the re-birth of deep learning based approaches (Bengio et al., 2003), many researchers have applied them to the twitter sentiment analysis with varying degrees of success(Le and Mikolov, 2014; Severyn and Moschitti, 2015a; Kalchbrenner et al., 2014; Severyn and Moschitti, 2015b; Johnson and Zhang, 2015; Socher et al., 2013; Poria et al., 2015; Tang et al., 2015). A top-performing system for SemEval2016, SwissCheese(Deriu et al., 2016), is based on convolutional neural networks (CNN). Our approach also proposes use of CNN for twitter sentiment classification. However, our work differs from others in using three novel semantics-based distributed representation features. Our contributions summarized into the following:

1. **A new method to learn sentiment-specific multi-word expressions (SSMWE) embeddings whose pre-trained vectors were used for sentiment classification, is put forward.**

   Strictly speaking, unsupervised learning of distributed representations (word embeddings) obviates the need for careful feature engineering; however, task-agnostic learning is unsuitable for specific NLP applications. E.g., distance between word embeddings of *happy* and *sad* will be small since they share similar contexts even though these words connote opposite sentiment polarity and thereby affecting the performance of sentiment classification adversely. (Tang et al., 2014b) proposed learning sentiment-specific word embedding (SSWE) for twitter sentiment classification. However, ignoring the structures of bigrams and trigrams undermines the linguistic aspects of data. Noticing that **multi-word expressions**(MWEs)(Sag et al., 2002)(e.g., *kick the bucket*,*shoot the breeze*) are independent semantic units whose interpretations cross word boundaries and their copious usage in tweets, our proposal learns distributed representations of MWEs jointly encoding their sentiment polarity into them.

2. **Accommodate (Bartunov et al., 2016)'s approach to learn multi-prototype word embeddings and use sense-disambiguated word embeddings (SDWE).**

   Indeed, existing approaches to learn unsupervised distributed representations use massive text datasets to capture syntactic/semantic contexts of words. While such representations achieve algebraic semantic compositionality, they are unsuitable for the NLP applications that require finer semantic processing. An obvious short-coming of such methods is in failing to distinguish representations for different senses of the same word (**polysemy**). For example, the word *bank* may refer to a *financial organization* or to a *river bank* depending on the context; but since these methods learn a *unique representation* for each word, either the most frequent meaning of the word dominates the other senses or the meanings are mixed in the vector representations.

3. **Augment the feature space by including paradigmatic similarity information from WordNet through distributed representations**

   Indeed, distributed representations of words trained from massive text datasets are acknowledged to capture well the syntagmatic relations between the words and therefore serve as a faithful representation of similarity in the feature space. However, methods that learn such representations ignore manually curated ontologies which can provide additional semantic information between words (e.g., paradigmatic relations). We propose to "augment" distributed word representations by hypernym and hyponym word embeddings and thereby making such representations more informative. Finding hypernyms and hyponyms of a word entails finding the correct sense of that word and therefore requires word-sense disambiguation(WSD). Using the approach by (Bartunov et al., 2016), we learn multi-prototype distributed word representations and perform WSD using Dirichlet-process based

model.

The rest of the paper is organized as follows: In Section 2, we discuss the work closely related to ours; Section 3 depicts our overall approach to twitter sentiment classification; Section 4 discusses our features – especially, our model of learning SSMWE embeddings. In Section 5, we present the details of our CNN model for twitter sentiment classification. Section 6 explains our experimental setup and results compared with other competing approaches followed by error analysis. Finally, in Section 7, we present our conclusions.

2 RELATED WORK

With a view to the increasing applications of twitter sentiment analysis, recent years have seen a very rapid growth of research in this area. Below we describe the previous work that is closely related to our work: (a) applying deep learning models for twitter sentiment analysis, and (b) using novel word embedding features in sentiment classification.

2.1 Twitter Sentiment Classification using Deep Learning Models

Previous approaches of task-specific feature engineering are replaced with deep neural networks that show promise at capturing salient features automatically in a supervised or unsupervised setup (Collobert et al., 2011). Following (Kim, 2014)'s architecture that uses multiple filters with varying window sizes that are applied on each given sentence, (Le and Mikolov, 2014; Severyn and Moschitti, 2015a; Kalchbrenner et al., 2014; Severyn and Moschitti, 2015b; Johnson and Zhang, 2015) all show success of Convolutional Neural Networks (CNN) for sentiment analysis task. While (Severyn and Moschitti, 2015a) propose a 1-layer architecture, (Deriu et al., 2016) uses 2 hidden layers to boost better feature learning. (Kalchbrenner et al., 2014) proposed Dynamic Convolutional Neural Network which they show outperforms other unigram and bigram based methods on classification of movie reviews and tweets. Other neural network architectures have also demonstrated good performance for sentiment analysis task; particularly, (Socher et al., 2013)'s recursive neural tensor network (RNTN), (Tang et al., 2015)'s long short term memory (LSTM) network. (Poria et al., 2015) use CNN to learn a 306 dimensional vector consisting of word embedding and part of speech values and use it with their multiple-kernel approach for multimodal sentiment analysis.

2.2 Novel Word Embedding Features used for Twitter Sentiment Classification

(Mohammad et al., 2013) achieve best results at SemEval2013 twitter sentiment classification using hand-crafted features. They also use several lexicons to determine the sentiment score for each token in the tweet, part-of-speech tag and hashtag. Following (Harris, 1954)'s distributional hypothesis ("linguistic items with similar distributions have similar meanings"), ever since (Bengio et al., 2003) proposed learning unsupervised pre-trained distributional word representations, several complex NLP tasks use such word embeddings as features. The central idea is to jointly learn an embedding of words into a low dimensional dense vector space. The word vectors inside the embedding matrix capture distributional syntactic and semantic information via the words co-occurrence statistics. Realizing the limitations of bag-of-word one-hot vector representation in classification such as sparse high-dimensional vector space and lack of capturing semantic relatedness of words, several researchers (Mikolov et al., 2013a; Pennington et al., 2014; Collobert et al., 2011), proposed learning word embeddings in unsupervised setup. Word2Vec(Mikolov et al., 2013b) uses two frameworks: Skip-Gram and Continuous Bag-of-Words (CBOW). CBOW uses a words context words in a surrounding window to predict the word, while Skip-Gram uses a word to predict its surrounding words.

Several researchers have attempted to improve learning word embeddings for sentiment analysis. (Tang et al., 2014b) modify (Collobert et al., 2011)'s method to learn sentiment- specific word embeddings (SSWE) from massive distant-supervised tweets. (Liu et al., 2015) extends Skip-Gram by treating topical information as important *a priori* knowledge for training word embedding and proposes the topical word embeddings (TWE) model, where words and their affiliated topic derived from Latent Dirichlet Allocation (LDA) are combined to obtain the embedding.(Ren et al., 2016) also follows similar approach. (dos Santos and Zadrozny, 2014) proposed a neural network architecture that exploits character-level, word-level and sentence-level representations. Character-level features proved to be useful for sentiment analysis on tweets, because they capture morphological and shape information. (Labutov and Lipson, 2013) re-embed the words from existing word embeddings for supervised sentiment classification.

Unlike traditional methods to learn a unique representation for each word, (Huang et al., 2012; Reisinger and Mooney, 2010) propose various neural

network-based methods for learning multi-prototype representations. Recently, various modifications of Skip-gram(Mikolov et al., 2013b) are proposed to learn multi-prototype representations. (Qiu et al., 2014) propose proximity-ambiguity sensitive Skip-gram for each Part-of-speech of a given word. However, a word can still have multiple meanings even with the same part-of-speech tag (e.g., *crane*) which is not addressed by them. (Tian et al., 2014) provides improvement over original Skip-gram but it is not clear how to set the number of prototypes. (Chen et al., 2014) proposes to learn single-prototype representations with Skip-gram and later uses WordNet to learn multi-prototype representations for ambiguous words. (Bartunov et al., 2016)'s Adaptive Skip-gram (AdaGram) model does not consider any form of supervision and learns the sense inventory automatically from the raw text. (Neelakantan et al., 2015) proposed Multi-sense Skip-gram (MSSG) and non-parametric (NP MSSG) which fixes the number of prototypes a priori similar to (Tian et al., 2014) and uses greedy procedure that allocates new representation for a word if existing ones explain its context below some threshold.

Overall, we find that the approach proposed by (Bartunov et al., 2016) is the most generic and fast compared to others. Not only it allows to efficiently learn required number of prototypes for ambiguous words, but is able also to gradually increase the number of meanings when more data becomes available thus distinguishing between shades of same meaning.

3 APPROACH

Figure 1 depicts the architecture for our twitter sentiment classification model. Two independent offline components SSMWE and SDWE take as input a massive tweet database and generates feature vectors to be used in the classification model. SSMWE uses distant-supervised tweet database and finds occurrences of MWE in tweets through lookup in WikiMWE dictionary and learns sentiment-encoded distributed representations. SDWE learns multi-prototype word representations in unsupervised fashion and also outputs a model for the subsequent word-sense disambiguation. Corresponding to the correct sense of the word, we use *word2vec* embeddings for the hypernym and hyponym words as additional features (we call this WNE). In addition to these SSMWE, SDWE, and WNE features, we also use hand-crafted features to train our CNN model for twitter sentiment classification.

The details of our novel semantics-based distributed representation features are presented in Section 4.

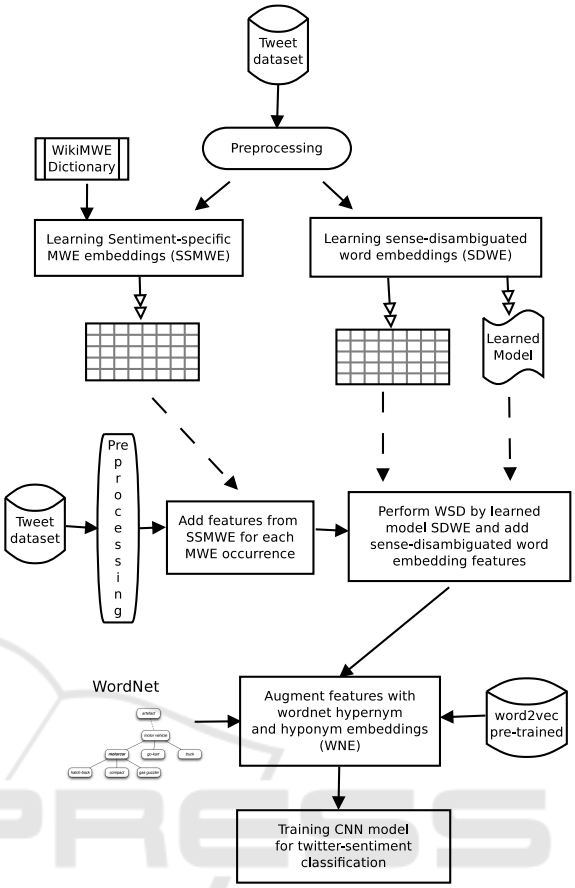

Figure 1: Our approach to twitter sentiment classification.

4 FEATURES

4.1 SSMWE Embeddings

Traditional methods of learning word embedding(Collobert et al., 2011; Pennington et al., 2014) employ an unsupervised setup, by modeling syntactic contexts of words but ignoring sentiment information. However, such methods cannot distinguish words with similar context but opposite sentiment polarity. E.g., distance between vector embeddings of the words *happy* and *sad* will be small since they share similar contexts even though these words connote opposite sentiment polarity; thereby, affecting the performance of sentiment classification adversely. (Tang et al., 2014b) propose learning sentiment-specific word embedding (SSWE) under a supervised learning framework by integrating sentiment information into the loss functions that the

model tries to minimize. However, they empirically show that vector embeddings for bigrams and trigrams do not improve the performance of twitter-sentiment classification. We believe this is due to learning *all* bigrams and trigrams instead of specific ones which carry specific semantics such as *Multi-word expressions (MWE)*. Usage of MWEs in language is quite prevalent; yet, only a few NLP applications take cognizance of it. We propose to learn sentiment-specific MWE embeddings (SSMWE) by finding syntactic context around MWE occurrence using a model similar to (Tang et al., 2014b).

Learning SSMWE Embeddings

To find sentiment-specific embeddings, (Tang et al., 2014b) propose to integrate the sentiment information by predicting the sentiment distribution of text based on input ngram simultaneously while finding word embeddings. Given an original (or corrupted) ngram and the sentiment polarity of a sentence as the input, their model predicts two scores for each input ngram. The two scalars f_{LM} and f_{SS} stand for language model score and sentiment score of the input ngram, respectively. Language model score measures the strength of correct learning of word embedding based on the syntactic contexts of words, and sentiment score evaluates the correct sentiment polarity prediction. Our model for learning SSMWE embeddings is based on (Tang et al., 2014b; Collobert et al., 2011). However, instead of learning single word embeddings, we propose to learn the sentiment-polarity encoded embeddings of multi-word expressions. To identify occurrences of MWEs, we use WikiMWE(Hartmann et al., 2012) which contains over 350,000 multi-word expressions mined from Wikipedia. For example, in our architecture as shown in figure 2, the phrase *shooting the breeze* in the input sentence is identified as an MWE.

The model comprises of the following layers:

1. Lookup layer: this layer maps the word/MWE identifiers to the embedding vectors.
2. Hidden layer: this is fully connected (represented by matrix M_1) with the Lookup layer and outputs *hardtanh* values.
3. Linear layer: this generates the two different scores – language model score, and sentiment classification score by two different linear models represented by M_2^{LM} and M_2^{SS} matrices.

The overall equation of the model is:

$$f_{LM}(\mathbf{t}) = M_2^{LM} \cdot htanh(M_1 \times \mathbf{t} + b_1) + b_2^{lm} \quad (1)$$

$$f_{SS}(\mathbf{t}) = M_2^{SS} \cdot htanh(M_1 \times \mathbf{t} + b_1) + b_2^{ss} \quad (2)$$

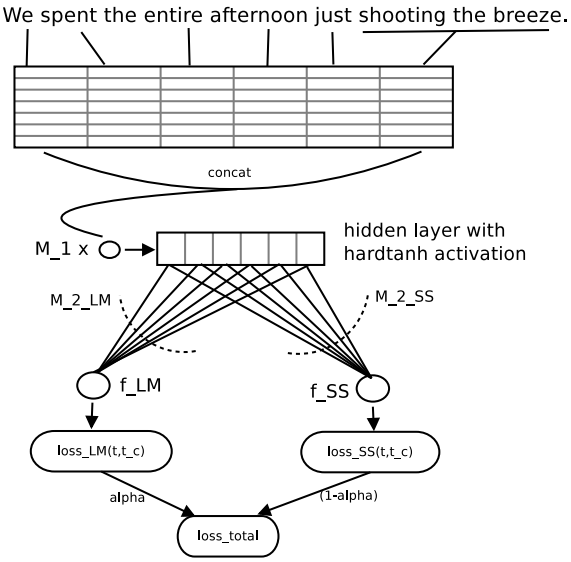

Figure 2: Neural network to learn SSMWE embeddings.

where the two scalars $f_{LM}(\mathbf{t})$ and $f_{SS}(\mathbf{t})$ are language model score and sentiment score of the input ngram $\mathbf{t}$, respectively; $\mathbf{t}$ is input n-gram which is concatenation of m word/MWE embeddings $x \in R^{d\times 1}$; $M_1 \in R^{h\times(d\times m)}$ and $b_1 \in R^{h\times 1}$ are parameters of the hidden layer; $M_2^{LM}, M_2^{SS} \in R^{h\times 1}$ and $b_2^{lm}, b_2^{ss} \in R$ are parameters of the output layer.

Traditional word embedding methods work by introducing a loss function which computes a loss between an original input n-gram and "corrupted" n-gram which is produced by replacing a word from the input n-gram randomly. The training objectives are: (1) the original n-gram should obtain a higher language model score $f_{LM}(\mathbf{t})$ than the corrupted ngram $f_{LM}(\mathbf{t^c})$, and (2) the sentiment score of original ngram $f_{SS}(\mathbf{t})$ should be more consistent with the gold polarity annotation of sentence than corrupted ngram $f_{SS}(\mathbf{t^c})$.

The loss function is the linear combination of two hinge losses,

$$loss_{tot}(t,t^c) = \alpha \cdot loss_{LM}(t,t^c) + (1-\alpha) \cdot loss_{SS}(t,t^c) \quad (3)$$

where $loss_{LM}$ and $loss_{SS}$ are the loss functions for language model and sentiment polarity score respectively.

The $loss_{LM}$ is defined as under:

$$loss_{LM}(t,t^c) = max(0, 1 - f_{LM}(t) + f_{LM}(t^c)) \quad (4)$$

where t is the original ngram, t^c is the corrupted ngram, $f_{LM}(\cdot)$ is the language model score of the input n-gram.

The $loss_{SS}$ is defined as under:

$$loss_{SS}(t,t^c) = max(0, 1 - \delta_s(t) f_{SS}(t) + \delta_s(t) f_{SS}(t^c)) \quad (5)$$

where $f_{SS}(t)$ is the predicted sentiment score of the input n-gram and $\delta_s(t)$ is defined as under:

$$\delta_s(t) = \begin{cases} 1, \text{ if } f^g(t) = [1,0] \\ -1 \text{ if } f^g(t) = [0,1] \end{cases} \quad (6)$$

We train sentiment-specific MWE embedding from Setiment140 corpus(Go et al., 2009) which contains massive distant-supervised tweets collected with positive and negative emoticons. The corpus has about 1.6 million tweets, half of which are with positive sentiment polarity and the remaining half with negative polarity. Unlike (Tang et al., 2014b), who use AdaGrad(Duchi et al., 2011) to update the parameters, we use Adam optimization(Kingma and Ba, 2014). We empirically set the window size as 3, the size of embedding as 100 and number of hidden layer neurons as 30. Learning rate was set as 0.1.

4.2 Sense-disambiguated Word Embeddings (SDWE)

Most prior work on learning word representations do not take into account word ambiguity and maintain only a single representation per word. The word *apple* may refer to a *fruit* or to the *Apple Inc.* depending on the context. Popular approaches to learn word embeddings such as Socher's Continuous Bag of Words (CBOW) and SkipGram and Collobert's model also fail to address this *polysemous* nature of word usage and find a unique representation for each word. NLP applications needing finer semantic processing (such as sentiment analysis) entail finding the embedding for the correct sense of the word in the context. Among a few approaches proposed in the past [sec. 2.2], we choose the method proposed by (Bartunov et al., 2016). Their model is a nonparametric Bayesian extension of Skip-gram. We use their approach and use their toolkit available on github to train the Adaptive SkipGram model on Sentiment140 dataset to learn multiple representations of words, one per sense. After the model is trained, we predict/induce the correct sense of each word using their model based on Bayesian non-parametrics – Dirichlet process infinite mixture model. We use the word embedding corresponding to this correct sense of the word rather than using generic C&W vector embeddings.

Table 2: Additional features used for twitter sentiment classification. B=boolean, N=integer, R=real number.

Feature	**Value**
Relative position of the word from the first word	N
Relative position of the word from the nearest word denoting negation	N
Whether the word starts with a capitalized letter	B
Whether the word contains repeated vowels (e.g. *goooood*, *coooool*)	B
Presence of positive emoticon	B
Presence of negative emoticon	B
Sentiment score of the word in sentiment lexicons	R
Whether the word denotes negation (e.g. *no*, *never*)	B
Whether the word is a punctuation mark	B

4.3 WordNet Features(WNE)

The word vector embeddings capture syntagmatic relations between words but may fail to capture paradigmatic relations for which we use WordNet. We use the word embeddings of the immediate hypernym and hyponym of the correctly disambiguated sense of the given word as features in supervised classification. After learning multi-prototype word representations using the method proposed by (Bartunov et al., 2016) as above and finding the correct word embedding correspoding to the correct sense of the word used in the context, we find the immediate hypernyms and hyponyms of that sense of that word and use *word2vec* embeddings of these words as additional features to train our model.

4.4 Hand-crafted Word Features

Despite the obvious advantages of distributed representations, there are a few discriminatory features missed. We augment our vector embeddings with the following feature vector:

5 CNN MODEL FOR TWITTER SENTIMENT CLASSIFICATION

Figure 3 illustrates our convolutional neural network for twitter sentiment classification. Our architecture is very similar to (Kim, 2014; Deriu et al., 2016). Features for our model: 1. sentiment-specific MWE embeddings, 2. sense-disambiguated word vectors, 3. WordNet features, and 4. hand-crafted word features. All these are explained in the section above.

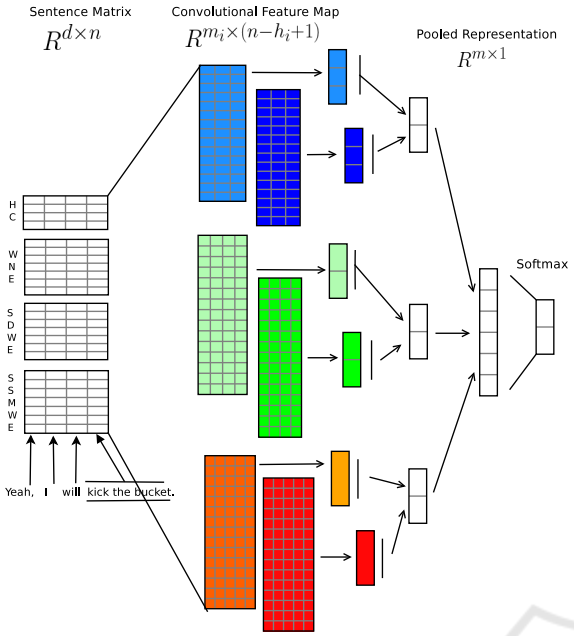

Figure 3: Convolutional Neural Network model for twitter sentiment classification. SSMWE: sentiment-specific MWE features, SDWE: sense-disambiguated word embeddings, WNE: WordNet features, HC: hand-crafted features.

The details of the layers of the CNN architecture are as follows:

1. **Sentence Matrix** A tweet is represented by horizontal concatenation of d-dimensional word embeddings of its n constituent tokens. The tokens can be any MWE present in the tweet or individual words. This generates a matrix $S \in R^{d\times n}$ which is input to the convolutional neural network model.
2. **Convolutional Layer** Convolution layer comprises of multiple filters of fixed length which are convolved with the input sentence matrix to extract discriminative word sequence patterns useful for classification. The convolution operation is defined as under:

$$c_i = \sum_{k,j} (S_{[i:i+h]})_{k,j} \cdot F^m_{k,j} \tag{7}$$

where S is input sentence matrix, h is filter width, and $F^m_{k,j}$ are m^{th} filter's coefficients. c_i is the value of the learned feature. The entire convolution of the m^{th} filter with the input tweet produces $n-h+1$ values which are concatenated together to produce a vector $\mathbf{c} \in R^{n-h+1}$. The vectors $\mathbf{c}$ are then aggregated over all m filters into a feature map matrix $C \in R^{m\times(n-h+1)}$.

3. **Max Pooling** The output of the convolutional layer is passed through a non-linear activation function such as *hardTanh* or *sigmoid* or *ReLU*. Pooling layer aggregates vector elements by taking the maximum from each element of the convolutional feature map. The resulting vector is $\mathbf{C_{pooled}} \in R^{m\times 1}$.
4. **Softmax** Pooling layer output $\mathbf{C_{pooled}} \in \mathbf{R^m}$ is used for softmax regression which returns the class $\hat{y} \in [1,K]$ with largest probability. i.e.,

$$\hat{y} = \arg\max_j P(y=j \mid \mathbf{x},\mathbf{w},\mathbf{a}) \tag{8}$$

$$q = \arg\max_j \frac{e^{(\mathbf{C_{pooled}}\mathbf{w_j}+a_j)}}{\sum_{k=1}^{K} e^{(\mathbf{C_{pooled}}\mathbf{w_j}+a_j)}} \tag{9}$$

where $\mathbf{w_j}$ denotes the weights vector of class j and a_j the bias of class j.

- **Model Parameters** The objective of training the model over a dataset of tweets is to learn the following parameter set: $S, F, \mathbf{W}, \mathbf{a}$. Where S is a sentence matrix consisting of word embeddings, F are filter weights, $\mathbf{W}$ are concatenation of the weights $\mathbf{w_j}$ for every output class in the softmax layer, and $\mathbf{a}$ the bias of the softmax layer. The loss is minimized using the Adam optimizer(Kingma and Ba, 2014).
- **Regularization** We use the dropout method proposed by (Srivastava et al., 2014) after the max pooling layer: each dimension is randomly set to 0 using a Bernoulli distribution $B(p)$ where p is a hyperparameter. In addition, we complement this method of regularization with $L2$-Regularization of softmax parameters.

6 EXPERIMENTS AND RESULTS

6.1 Dataset and Experimental Setup

To evaluate our novel features for twitter sentiment classification, we use supervised setup of convolutional neural networks similar to (Kim, 2014). We perform experiments on the benchmarked dataset from SemEval sentiment analysis shared task (from 2013 through 2016). Since the datasets available on SemEval website contain only tweet identifiers, we used Twitter API to download the actual tweets. However, some of the tweets could not be downloaded as they may have been deleted. Table 3 summarizes the dataset of those tweets that we could obtain using the API.

Baseline Methods

We compare our method with three top performing systems whose methodology is very close to our ap-

Table 3: SemEval dataset summary. Txx stands for Twitter20xx and S13 stands for SMS2013.

	Pos	Neg	Neutral	All
T13-train	3,641	1,457	4,586	9,684
T13-dev	575	340	739	1,654
T13-test	1,475	559	1,513	3,547
S13-test	492	394	1,208	2,094
T14-test	982	202	669	1,853
LJ14-test	427	304	411	1,142
T15-test	1,038	365	987	2,390
T16-train	3,094	2,043	863	6,000
T16-dev	843	391	765	1,999
T16-devtest	994	325	681	2,000
T16-test	7,059	10,342	3,231	20,632
Total	**20,620**	**16,722**	**15,653**	**52,995**

proach. Since we could not find their original implementations, we re-implement these methods with the choices of the hyperparameters for which they obtained best results:

- NRC-canada (Mohammad et al., 2013): Top ranked in SemEval2013 twitter sentiment classification task, their system uses diverse sentiment lexicons and hand-crafted features for training SVM classification model.
- Coooolll (Tang et al., 2014a): ranked 2nd on the Twitter2014 test set of SemEval 2014 Task 9, Coooolll employs SVM for twitter sentiment classification using their sentiment-specific word embedding (SSWE)(Tang et al., 2014b) features in addition to using features from (Mohammad et al., 2013). They learn SSWE from 10M tweets using emoticons in a distant supervision model.
- SwissCheese (Deriu et al., 2016): Top ranked in SemEval-2016 twitter sentiment classification task (Task 4), SwissCheese leverages large amounts of data with distant supervision to train an ensemble of 2-layer convolutional neural networks whose predictions are combined using a random forest classifier. However, unlike them, we did not train the model on 90M tweets in the distant-supervised phase in the first epoch.

Preprocessing

We perform the following steps sequentially as preprocessing:

1. Remove tweets that are too short (i.e., less than 6 words)
2. Remove @user, URLs, and hashtags from each tweet as they directly do not contain sentiment information.
3. We use the Stanford tokenizer[4] to tokenize the tweets.
4. Replace all occurences of MWEs by unique identifiers. We use WikiMWE(Hartmann et al., 2012) which contains multiword expressions mined from Wikipedia for finding occurrences of MWEs in tweets. It contains over 350,000 multiword units of size 2-4, including technical terminology, non-compositional multiword expressions, and collocations. For example, in the sentence "Yeah, i will kick the bucket today", we find that the phrase *kick the bucket* is present in the WikiMWE lexicon and so we replace it with *ktb*001.
5. Remove stop words[5]. To find stop words, we temporarily convert the tweet words in lower case; however, for subsequent processing, we keep the words in their original case.
6. Use of words like *cooooolll*, *awesommmme*, are sometimes used in tweets to emphasize emotion. We use a simple trick to normalize such occurrences. Let n denote the number of such letters that have three or more consecutive occurrences in a given word. We first replace three or more consecutive occurrences of the same character with two occurrences. Then we generate $\binom{n}{2}$ prototypes that are at edit distance 1 (only delete operation, deleting only repeated character) and look for this prototype in the dictionary to find the word. For example, *coooooollll* $\rightarrow$ *cooll* $\rightarrow$ *cool*.
7. We use an acronym dictionary from an online resource[6] to find expansions of the tokens such as *gr8*, *lol*, *rotfl*, etc.
8. Assign a unique *vocabulary id to each token. At testing time, we ignore the unknown words.*

Experimental Setup

- To find the prior polarity of words, we use SentiWordNet 3.0(Baccianella et al., 2010). SentiWordNet is a pre-trained lexical resource that assigns to each synset of WordNet 3.0 three sentiment scores: positivity, negativity, neutrality. Since we perform WSD, using SentiWordNet provides us with better sentiment scores than other lexicons such as MPQA, AFINN and Bing Lius Opinion Lexicon.

[4]https://nlp.stanford.edu/software/tokenizer.htm
[5]http://www.nltk.org
[6]http://www.noslang.com

- We use DeepNL(Attardi, 2015) toolkit to learn sentiment-specific multiword expressions[7]. DeepNL implements (Tang et al., 2014b) and provides code for creating word embeddings from text. We use a public dataset Sentiment140(Go et al., 2009) which used distant supervision approach (using emoticons to generate sentiment label). Sentiment140 contains 1.6M tweets with positive and negative labels.
- To learn multi-prototype word representations we use (Bartunov et al., 2016)'s Adaptive Skip-gram (AdaGram) model and their code from an online git repository[8].
- For implementing CNN, we have used TensorFlow[9] and also borrow code from an online github repository[10]. We use Sentiment140 for learning multi-prototype representations.
- To augment WordNet hypernym/hyponym features, we use word2vec[11] vectors which has vocabulary of 3 million words and phrases. These publicly available vectors have dimensionality of 300 and are trained on roughly 100 billion words from a Google News dataset using the continuous bag-of-words architecture(Mikolov et al., 2013b).
- We conducted our experiments on Intel Core i5 machine (4 cores), with 16 GB RAM.

Hyperparameters

- For learning SSMWE embeddings (DeepNL): size of word window: 5, embedding vector size: 100, learning rate: 0.05, number of hidden neurons: 200.
- For CNN classification model: filter sizes: 3,4,5, number of filters per filter size: 128 filters per filter size, learning rate: 0.001, activation unit: ReLU, dropout probability: 0.4.
- For finding multi-prototype representations (AdaGram): size of word window: 6, embedding vector size: 200, maximum number of learned prototypes: 3, parameter for Dirichlet process α: 0.1, minimum word frequency below which a word will be ignored: 20.

[7]https://github.com/attardi/deepnl

[8]https://github.com/sbos/AdaGram.jl

[9]https://www.tensorflow.org/

[10]https://github.com/bernhard2202/twitter-sentiment-analysis

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

6.2 Results and Error Analysis

Table 4 shows the results on the SemEval test datasets. It shows the comparison of macro-averaged F1 score (for all three classes: positive, negative, neutral) we obtain using our CNN model trained with baseline features and newly introduced features. SSWE features use (Tang et al., 2014b)'s sentiment-specific word embeddings, SSMWE uses our sentiment-specific MWE embeddings. Each subsequent row of the table uses additional features with SSMWE features. SDWE uses sense-disambiguated word embeddings, HC uses hand-crafted features, and WNE uses word2vec embeddings of the hypernym and hyponym of each word. Results are displayed on the combined set of tweets in the training, testing, and development sets for each SemEval dataset. The last column in table (δ) shows the incremental percentage improvement each from the previous step starting with SSWE (baseline).

Table 4: Effect of our features on SemEval dataset. Overall 14.24% improvement from baseline.

Features ↓	S13-test	LJ14-test	T13-test	T14-test	T15-test	T16-test	δ
word2vec	51.76	53.09	50.44	50.58	58.87	60.89	—
SSWE	62.81	70.04	69.76	63.56	65.33	61.71	0
SSMWE	70.43	70.33	71.66	68.45	71.12	70.04	**+7.32%**
+SDWE	72.11	71.69	74.11	72.53	73.76	72.67	**+3.51%**
+HC	**75.02**	**73.87**	**75.38**	**77.13**	**75.11**	**75.17**	**+3.41%**
+WNE	69.78	69.34	70.06	65.89	66.33	65.57	-10.1%

Table 5 compares our best best SSMWE model with baseline models. NRC-canada is our implementation of (Mohammad et al., 2013) which uses hand-crafted features and sentiment scores from many online sentiment lexicons; Cooolll is our implementation of (Tang et al., 2014a) which uses SSWE; SwissCheese is our implementation of (Deriu et al., 2016)

Table 6 presents the confusion matrix with respect to all consolidated SemEval data with our SSMWE+SDWE+HC features.

Table 7 presents the effect on Macro-F1 score by various combinations of features. For example, column 2 of row 2 shows the results with SSMWE+HandCrafted features, and column 3 or row 2 are the results of SSMWE+HandCrated+WordNetEmbeddings features.

Table 5: Macro-F1 comparison of baseline with our model on SemEval test datasets.

Model	S13-test	LJ14-test	T13-test	T14-test	T15-test	T16-test
NRC-Canada	66.23	70.34	67.51	68.13	69.12	70.30
Coooolll	67.68	72.90	70.40	70.14	71.09	68.23
Swiss-Cheese	67.81	67.19	68.01	68.95	64.75	62.83
SSMWE Best	**74.43**	**74.44**	**77.33**	**76.39**	**75.21**	**74.87**

Table 6: Confusion matrix with respect to all consolidated SemEval test data with our SSMWE+SDWE+HandCrafted features.

	Positive	Negative	Neutral
Positive	7815	933	2724
Negative	1322	3334	405
Neutral	1657	2044	11424

Table 7: Effect of various feature combinations on macro-F1 score on consolidated SemEval dataset.

SSMWE	+SDWE	+HCraft	+WNE
+SDWE	72.29	**75.46**	67.87
+HCraft	—	69.17	70.91
+WNE	—	—	62.61

Discussion

As can be seen from the results, our novel semantics based distributed features significantly outperform baseline. Table 4 shows the incremental percentage improvement in accuracy starting with SSWE. Rather than using all unigram and bigram embeddings, using only MWE embeddings (which are sentiment-encoded) increases the macro-averaged F1 score by 7.32%. We attribute this to the fact that MWEs are independent semantics-bearing units so that the use of distributed representations learned in sentiment-distinguishing supervised model provides better discriminative power over ngram word embeddings. Adding sense-disambiguated word embeddings (SDWE) improves the score further by 3.51% totaling 10.83% improvement over baseline. While word2vec vectors learned from a massive dataset exhibits algebraic semantic compositionality, they are found to be less suitable for tasks requiring finer semantic processing since they ignore different meaning of the same word. On the other hand, because of their use of distributed representations of the correct sense of each word, SDWE seems intuitively appealing features for the underlying classification model and achieves higher accuracy score. Adding hand-crafted features further increase the score by 3.41% **totaling 14.24% overall improvement**. The reason for this is in our view largely because of two particular hand-crafted features: negation, and prior polarity from existing sentiment lexicons. Lastly, we note that using WordNet hypernym and hyponym words embeddings degrade the performance. This is owing to the fact that including hypernym and hyponym embeddings "dilutes" specificity of the feature. We undertake a more principled investigation of this in our future work.

It is observed from Table 6 that misclassifications between positive and neutral categories are more frequent than other classes. We believe this is due to the limitation of our model in understanding "objective" polarity of the tweet given the words present. However, we also note that there are many errors in gold-standard human evaluations too with respect to this. Another observation is the difficulty in achieving precision in negative class. This is due to the *sarcasm* and *idioms* used to convey negativity.

Error Analysis

Throughout the development and testing of our developed approach, we noticed several cases where the errors are inevitable regardless of the efficiency and soundness of the model because of either lack of discriminating information or the subjectivity of the statement that would make any sound mathematical model void. Examples of such non-avoidable errors are listed below:

- Errors due to presence of equal number of words with opposing polarity. For instance, our method misclassified the tweet: *'Girls Gone Wild' creator Joe Francis hit with $20 million verdict in Steve Wynn lawsuit...* as positive due to presence of words "gone wild" and "verdict" even though the polarity of the tweet is negative.
- Errors due to unseen words and phrases. E.g., the tweet *Why these mfs jus blastin Rella like itz the hood on Saturday N shit lolz smh* is misclassified as positive due to the word 'lolz'. However most of the words in this tweet were unknown to the system.
- Errors due to non-standard way of writing a word/slang or unknown slang. E.g., the tweet *even the sun shines on a dogs a$$ some days ...* was misclassified as positive owing to the failure of our system to understand 'a$$'.
- Errors due to information loss from hashtag removal in preprocessing where many tweets have

crucial information about the sentiment in the hash-tag. For example, the tweet *Buzzed that Im going Amsterdam on Friday! Ajax tickets could do with turning up though #fuckyouviagogoext* is misclassified as positive in our system.

7 CONCLUSIONS

In this paper, we proposed three novel semantics-based distributed representation features for Twitter sentiment classification: sentiment-specific multi-word expressions embeddings (SSMWE), sense-disambiguated word embeddings (SDWE), and hypernym and hyponym embeddings (WNE) of the correct sense of a given word. Because of its acknowledged performance in past SemEval Sentiment Analysis competitions, we advocated the use of a convolutional neural network architecture. The performances of the newly introduced features are then compared to state of art model and baseline in SemEval benchmarked dataset, where an improvement of more than 14% has been testified. On the other hand, the results also showed that unlike the previous approaches to learn all bigrams and trigrams embeddings, learning specific (MWE) embeddings provides better discriminative features for classification. Further, we presented an approach to learn such distributed representations of MWE jointly encoding their sentiment polarity information into them. Motivated by our finding that *word2vec* embeddings learned from massive text datasets ignores polysemy and thereby fails to faithfully represent fine semantic information, we also advocate the use of sense-specific word embeddings for twitter sentiment classification.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their valuable suggestions because of which the technical quality of the work presented in this paper has improved.

REFERENCES

Attardi, G. (2015). Deepnl: a deep learning nlp pipeline. In *Proceedings of NAACL-HLT*, pages 109–115.

Baccianella, S., Esuli, A., and Sebastiani, F. (2010). Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Bartunov, S., Kondrashkin, D., Osokin, A., and Vetrov, D. (2016). Breaking sticks and ambiguities with adaptive skip-gram. In *Artificial Intelligence and Statistics*, pages 130–138.

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

Bollen, J., Mao, H., , and Zeng, X. (2011). Twitter mood predicts the stock market. In *Journal of Computational Science and 2(1) and pp. 1–8.*

Chen, X., Liu, Z., and Sun, M. (2014). A unified model for word sense representation and disambiguation. In *EMNLP*, pages 1025–1035.

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

Deriu, J., Gonzenbach, M., Uzdilli, F., Lucchi, A., Luca, V. D., , and Jagg, M. (2016). Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. In *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016) and San Diego and US*.

dos Santos, C. N. and Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Go, A., Bhayani, R., , and Huang, L. (2009). Twitter sentiment classification using distant supervision. In *CS224N Project Report*. Stanford.

Goodchild, M. F. and Glennon, J. A. (2010). Crowdsourcing geographic information for disaster response: a research frontier. In *International Journal of Digital Earth and 3(3) and pp. 231–241*.

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

Hartmann, S., Szarvas, G., and Gurevych, I. (2012). Mining multiword terms from wikipedia. In *Semi-Automatic Ontology Development: Processes and Resources*, pages 226–258. IGI Global.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). 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.

Johnson, R. and Zhang, T. (2015). Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in neural information processing systems*, pages 919–927.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Labutov, I. and Lipson, H. (2013). Re-embedding words. In *ACL (2)*, pages 489–493.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Liu, Y., Liu, Z., Chua, T.-S., and Sun, M. (2015). Topical word embeddings. In *AAAI*, pages 2418–2424.

Martinez-Camara, E., MartnValdivia, M. T., Lopez, L. A. U., , and Raez, A. M. (2014). Sentiment analysis in twitter. In *Natural Language Engineering and 20(1):128*.

Mejova, Y., Weber, I., , and Macy, M. W. (2015). Twitter: A digital socioscope. In *Twitter: A Digital Socioscope*. Cambridge University Press and Cambridge and UK.

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

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

Mohammad, S. M., Kiritchenko, S., and Zhu, X. (2013). Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., and Stoyanov, V. (2016). Semeval-2016 task 4: Sentiment analysis in twitter. *Proceedings of SemEval*, pages 1–18.

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

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Poria, S., Cambria, E., and Gelbukh, A. F. (2015). Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *EMNLP*, pages 2539–2544.

Qiu, L., Cao, Y., Nie, Z., Yu, Y., and Rui, Y. (2014). Learning word representation considering proximity and ambiguity. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Reisinger, J. and Mooney, R. J. (2010). 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.

Ren, Y., Wang, R., and Ji, D. (2016). A topic-enhanced word embedding for twitter sentiment classification. *Information Sciences*, 369:188–198.

Rosenthal, S., Nakov, P., Kiritchenko, S., Mohammad, S. M., Ritter, A., and Stoyanov, V. (2015). Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 451–463.

Rosenthal, S., Ritter, A., Nakov, P., and Stoyanov, V. (2014). Semeval-2014 task 9: Sentiment analysis in twitter. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 73–80. Dublin, Ireland.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., and Flickinger, D. (2002). Multiword expressions: A pain in the neck for nlp. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15. Springer.

Severyn, A. and Moschitti, A. (2015a). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962. ACM.

Severyn, A. and Moschitti, A. (2015b). Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*, pages 464–469.

Skoric, M., Poor, N., Achananuparp, P., Lim, E. P., , and Jiang, J. (2012). Tweets and votes: A study of the 2011 singapore general election. In *Proceedings of the 45th Hawaii International Conference on System Science (HICSS) and pp. 2583–2591*.

Smith, A. N., Fischer, E., and Yongjian, C. (2012). How does brand-related user-generated content differ across youtube and facebook and and twitter? In *Journal of Interactive Marketing 26(2) and pp. 102–113*.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., Potts, C., et al. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Tang, D., Qin, B., Feng, X., and Liu, T. (2015). Target-dependent sentiment classification with long short term memory. *CoRR, abs/1512.01100*.

Tang, D., Wei, F., Qin, B., Liu, T., and Zhou, M. (2014a). Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B. (2014b). Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565.

Tian, F., Dai, H., Bian, J., Gao, B., Zhang, R., Chen, E., and Liu, T.-Y. (2014). A probabilistic model for learning multi-prototype word embeddings. In *COLING*, pages 151–160.