A Joint Segmentation and Classification Framework for Sentiment Analysis

Duyu Tang[♮*], **Furu Wei**[♯], **Bing Qin**[♮], **Li Dong**[♯*], **Ting Liu**[♮], **Ming Zhou**[♯]
[♮]Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology, China
[♯]Microsoft Research, Beijing, China
[♯]Beihang University, Beijing, China
[♮]{dytang, qinb, tliu}@ir.hit.edu.cn
[♯]{fuwei, mingzhou}@microsoft.com [♯]donglixp@gmail.com

Abstract

In this paper, we propose a joint segmentation and classification framework for sentiment analysis. Existing sentiment classification algorithms typically split a sentence as a word sequence, which does not effectively handle the inconsistent sentiment polarity between a phrase and the words it contains, such as "*not bad*" and "*a great deal of*". We address this issue by developing a joint segmentation and classification framework (**JSC**), which simultaneously conducts sentence segmentation and sentence-level sentiment classification. Specifically, we use a log-linear model to score each segmentation candidate, and exploit the phrasal information of top-ranked segmentations as features to build the sentiment classifier. A marginal log-likelihood objective function is devised for the segmentation model, which is optimized for enhancing the sentiment classification performance. The joint model is trained only based on the annotated sentiment polarity of sentences, without any segmentation annotations. Experiments on a benchmark Twitter sentiment classification dataset in SemEval 2013 show that, our joint model performs comparably with the state-of-the-art methods.

1 Introduction

Sentiment classification, which classifies the sentiment polarity of a sentence (or document) as positive or negative, is a major research direction in the field of sentiment analysis (Pang and Lee, 2008; Liu, 2012; Feldman, 2013). Majority of existing approaches follow Pang et al. (2002) and treat sentiment classification as a special case of text categorization task. Under this perspective, previous studies typically use pipelined methods with two steps. They first produce sentence segmentations with separate text analyzers (Choi and Cardie, 2008; Nakagawa et al., 2010; Socher et al., 2013b) or bag-of-words (Paltoglou and Thelwall, 2010; Maas et al., 2011). Then, feature learning and sentiment classification algorithms take the segmentation results as inputs to build the sentiment classifier (Socher et al., 2011; Kalchbrenner et al., 2014; Dong et al., 2014).

The major disadvantage of a pipelined method is the problem of error propagation, since sentence segmentation errors cannot be corrected by the sentiment classification model. A typical kind of error is caused by the *polarity inconsistency* between a phrase and the words it contains, such as ⟨*not bad*, *bad*⟩ and ⟨*a great deal of*, *great*⟩. The segmentations based on bag-of-words or syntactic chunkers are not effective enough to handle the *polarity inconsistency* phenomenons. The reason lies in that bag-of-words segmentations regard each word as a separate unit, which losses the word order and does not capture the phrasal information. The segmentations based on syntactic chunkers typically aim to identify noun groups, verb groups or named entities from a sentence. However, many sentiment indicators are phrases constituted of adjectives, negations, adverbs or idioms (Liu, 2012; Mohammad et al., 2013a), which are splitted by syntactic chunkers. Besides, a better approach would be to utilize the sentiment information to improve the segmentor. Accordingly, the sentiment-specific segmentor will enhance the performance of sentiment classification in turn.

In this paper, we propose a joint segmentation and classification framework (**JSC**) for sentiment analysis, which simultaneous conducts sentence segmentation and sentence-level sentiment classification. The framework is illustrated in Fig-

[*] This work was partly done when the first and fourth authors were visiting Microsoft Research.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 477–487,
October 25-29, 2014, Doha, Qatar.

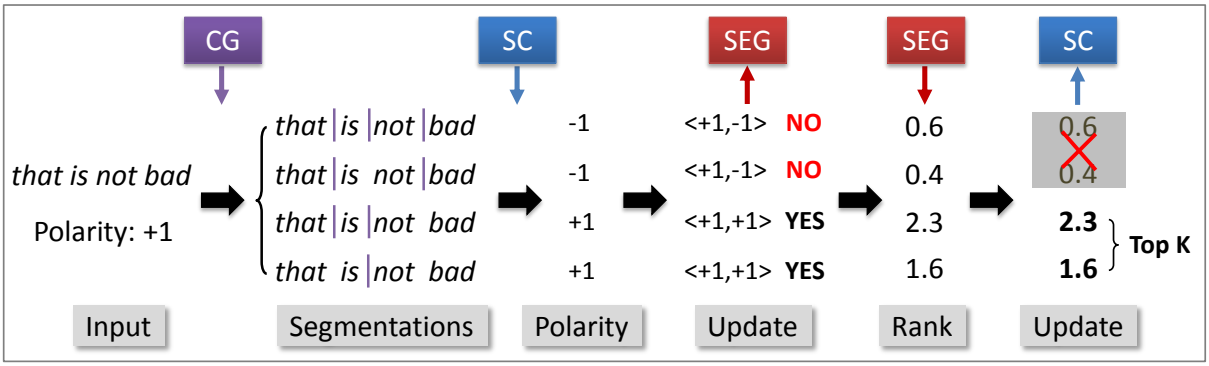


Figure 1: The joint segmentation and classification framework (**JSC**) for sentiment classification. **CG** represents the candidate generation model, **SC** means the sentiment classification model and **SEG** stands for the segmentation ranking model. **Down Arrow** means the use of a specified model, and **Up Arrow** indicates the update of a model.

ure 1. We develop (1) a candidate generation model to generate the segmentation candidates of a sentence, (2) a segmentation ranking model to score each segmentation candidate of a given sentence, and (3) a classification model to predict the sentiment polarity of each segmentation. The phrasal information of top-ranked candidates from the segmentation model are utilized as features to build the sentiment classifier. In turn, the predicted sentiment polarity of segmentation candidates from classification model are leveraged to update the segmentor. We score each segmentation candidate with a log-linear model, and optimize the segmentor with a marginal log-likelihood objective. We train the joint model from sentences annotated only with sentiment polarity, without any segmentation annotations.

We evaluate the effectiveness of our joint model on a benchmark Twitter sentiment classification dataset in SemEval 2013. Results show that the joint model performs comparably with state-of-the-art methods, and consistently outperforms pipeline methods in various experiment settings. The main contributions of the work presented in this paper are as follows.

- To our knowledge, this is the first work that automatically produces sentence segmentation for sentiment classification within a joint framework.

- We show that the joint model yields comparable performance with the state-of-the-art methods on the benchmark Twitter sentiment classification datasets in SemEval 2013.

2 Related Work

Existing approaches for sentiment classification are dominated by two mainstream directions. Lexicon-based approaches (Turney, 2002; Ding et al., 2008; Taboada et al., 2011; Thelwall et al., 2012) typically utilize a lexicon of sentiment words, each of which is annotated with the sentiment polarity or sentiment strength. Linguistic rules such as intensifications and negations are usually incorporated to aggregate the sentiment polarity of sentences (or documents). Corpus-based methods treat sentiment classification as a special case of text categorization task (Pang et al., 2002). They mostly build the sentiment classifier from sentences (or documents) with manually annotated sentiment polarity or distantly-supervised corpora collected by sentiment signals like emoticons (Go et al., 2009; Pak and Paroubek, 2010; Kouloumpis et al., 2011; Zhao et al., 2012).

Majority of existing approaches follow Pang et al. (2002) and employ corpus-based method for sentiment classification. Pang et al. (2002) pioneer to treat the sentiment classification of reviews as a special case of text categorization problem and first investigate machine learning methods. They employ Naive Bayes, Maximum Entropy and Support Vector Machines (SVM) with a diverse set of features. In their experiments, the best performance is achieved by SVM with bag-of-words feature. Under this perspective, many studies focus on designing or learning effective features to obtain better classification performance. On movie or product reviews, Wang and Manning (2012) present NBSVM, which trades-off

between Naive Bayes and NB-feature enhanced SVM. Kim and Zhai (2009) and Paltoglou and Thelwall (2010) learn the feature weights by investigating variants weighting functions from Information Retrieval. Nakagawa et al. (2010) utilize dependency trees, polarity-shifting rules and conditional random fields (Lafferty et al., 2001) with hidden variables to compute the document feature. On Twitter, Mohammad et al. (2013b) develop a state-of-the-art Twitter sentiment classifier in SemEval 2013, using a variety of sentiment lexicons and hand-crafted features.

With the revival of deep learning (representation learning (Hinton and Salakhutdinov, 2006; Bengio et al., 2013; Jones, 2014)), more recent studies focus on learning the low-dimensional, dense and real-valued vector as text features for sentiment classification. Glorot et al. (2011) investigate Stacked Denoising Autoencoders to learn document vector for domain adaptation in sentiment classification. Yessenalina and Cardie (2011) represent each word as a matrix and compose words using iterated matrix multiplication. Socher et al. propose Recursive Autoencoder (RAE) (2011), Matrix-Vector Recursive Neural Network (MV-RNN) (2012) and Recursive Neural Tensor Network (RNTN) (2013b) to learn the composition of variable-length phrases based on the representation of its children. To learn the sentence representation, Kalchbrenner et al. (2014) exploit Dynamic Convolutional Neural Network and Le and Mikolov (2014) investigate Paragraph Vector. To learn word vectors for sentiment analysis, Maas et al. (2011) propose a probabilistic document model following Blei et al. (2003), Labutov and Lipson (2013) re-embed words from existing word embeddings and Tang et al. (2014b) develop three neural networks to learn word vectors from tweets containing positive/negative emoticons.

Unlike most previous corpus-based algorithms that build sentiment classifier based on splitting a sentence as a word sequence, we produce sentence segmentations automatically within a joint framework, and conduct sentiment classification based on the segmentation results.

3 The Proposed Approach

In this section, we first give the task definition of two tasks, namely sentiment classification and sentence segmentation. Then, we present the overview of the proposed joint segmentation and classification model (**JSC**) for sentiment analysis. The segmentation candidate generation model and the segmentation ranking model are described in Section 4. The details of the sentiment classification model are presented in Section 5.

3.1 Task Definition

The task of sentiment classification has been well formalized in previous studies (Pang and Lee, 2008; Liu, 2012). The objective is to identify the sentiment polarity of a sentence (or document) as positive or negative [1].

The task of sentence segmentation aims to split a sentence into a sequence of exclusive parts, each of which is a basic computational unit of the sentence. An example is illustrated in Table 1. The original text "*that is not bad*" is segmented as "*[that] [is] [not bad]*". The segmentation result is composed of three basic computational units, namely *[that]*, *[is]* and *[not bad]*.

Type	**Sample**
Sentence	that is not bad
Segmentation	[that] [is] [not bad]
Basic units	[that], [is], [not bad]

Table 1: Example for sentence segmentation.

3.2 Joint Model (JSC)

The overview of the proposed joint segmentation and classification model (**JSC**) for sentiment analysis is illustrated in Figure 1. The intuitions of the joint model are two-folds:

- The segmentation results have a strong influence on the sentiment classification performance, since they are the inputs of the sentiment classification model.
- The usefulness of a segmentation can be judged by whether the sentiment classifier can use it to predict the correct sentence polarity.

Based on the mutual influence observation, we formalize the joint model in Algorithm 1. The inputs contain two parts, training data and feature extractors. Each sentence s_i in the training data

[1] In this paper, the sentiment polarity of a sentence is not relevant to the target (or aspect) it contains (Hu and Liu, 2004; Jiang et al., 2011; Mitchell et al., 2013).

Algorithm 1 The joint segmentation and classification framework (**JSC**) for sentiment analysis

Input:

training data: $T = [s_i, pol_i^g]$, $1 \leq i \leq |T|$
segmentation feature extractor: $sfe(\cdot)$
classification feature extractor: $cfe(\cdot)$

Output:

sentiment classifier: ***SC***
segmentation ranking model: ***SEG***

1: Generate segmentation candidates Ω_i for each sentence s_i in T, $1 \leq i \leq |T|$
2: Initialize sentiment classifier $\boldsymbol{SC}^{(0)}$ based on $cfe(\Omega_{ij})$, *randomize* $j \in [1, |\Omega_i|]$, $1 \leq i \leq |T|$
3: Randomly initialize the segmentation ranking model $\boldsymbol{SEG}^{(0)}$
4: **for** $r \leftarrow 1 \ldots R$ **do**
5: Predict the sentiment polarity pol_i for Ω_i based on $\boldsymbol{SC}^{(r-1)}$ and $cfe(\Omega_{i\cdot})$
6: Update the segmentation model $\boldsymbol{SEG}^{(r)}$ with $\boldsymbol{SEG}^{(r-1)}$ and $[\Omega_i, sfe(\Omega_{i\cdot}), \text{pol}_{i\cdot}, pol_i^g]$, $1 \leq i \leq |T|$
7: **for** $i \leftarrow 1 \ldots |T|$ **do**
8: Calculate the segmentation score for $\Omega_{i\cdot}$ based on $\boldsymbol{SEG}^{(r)}$ and $sfe(\Omega_{i\cdot})$
9: Select the top-ranked K segmentation candidates Ω_{i*} from Ω_i
10: **end for**
11: Train the sentiment classifier $\boldsymbol{SC}^{(r)}$ with $cfe(\Omega_{i*})$, $1 \leq i \leq |T|$
12: **end for**
13: $\boldsymbol{SC} \leftarrow \boldsymbol{SC}^{(R)}$
14: $\boldsymbol{SEG} \leftarrow \boldsymbol{SEG}^{(R)}$

T is annotated only with its gold sentiment polarity pol_i^g, without any segmentation annotations. There are two feature extractors for the task of sentence segmentation ($sfe(\cdot)$) and sentiment classification ($cfe(\cdot)$), respectively. The outputs of the joint model are the segmentation ranking model ***SEG*** and the sentiment classifier ***SC***.

In Algorithm 1, we first generate segmentation candidates Ω_i for each sentence s_i in the training set (line 1). Each Ω_i contains no less than one segmentation candidates. We randomly select one segmentation result from each Ω_i and utilize their classification features to initialize the sentiment classifier $\boldsymbol{SC}^{(0)}$ (line 2). We randomly initialize the segmentation model $\boldsymbol{SEG}^{(0)}$ (line 3). Subsequently, we iteratively train the segmentation model $\boldsymbol{SEG}^{(r)}$ and sentiment classifier $\boldsymbol{SC}^{(r)}$ in a joint manner (line 4-12). At each iteration, we predict the sentiment polarity of each segmentation candidate $\Omega_{i\cdot}$ with the current sentiment classifier $\boldsymbol{SC}^{(r-1)}$ (line 5), and then leverage them to update the segmentation model $\boldsymbol{SEG}^{(r)}$ (line 6). Afterwards, we utilize the recently updated segmentation ranking model $\boldsymbol{SEG}^{(r)}$ to update the sentiment classifier $\boldsymbol{SC}^{(r)}$ (line 7-11). We extract the segmentation features for each segmentation candidate $\Omega_{i\cdot}$, and employ them to calculate the segmentation score (line 8). The top-ranked K segmentation results Ω_{i*} of each sentence s_i is selected (line 9), and further used to train the sentiment classifier $\boldsymbol{SC}^{(r)}$ (line 11). Finally, after training R iterations, we dump the segmentation ranking model $\boldsymbol{SEG}^{(R)}$ and sentiment classifier $\boldsymbol{SC}^{(R)}$ in the last iteration as outputs (line 13-14).

At training time, we train the segmentation model and classification model from sentences with manually annotated sentiment polarity. At prediction time, given a test sentence, we generate its segmentation candidates, and then calculate segmentation score for each candidate. Afterwards, we select the top-ranked K candidates and vote their predicted sentiment polarity from sentiment classifier as the final result.

4 Segmentation Model

In this section, we present details of the segmentation candidate generation model (Section 4.1), the segmentation ranking model (Section 4.2) and the feature description for segmentation ranking model (Section 4.3).

4.1 Segmentation Candidate Generation

In this subsection, we describe the strategy to generate segmentation candidates for each sentence. Since the segmentation results have an exponential search space in the number of words in a sentence, we approximate the computation using beam search with constrains on a phrase table, which is induced from massive corpora.

Many studies have been previously proposed to recognize phrases in the text. However, it is out of scope of this work to compare them. We exploit a data-driven approach given by Mikolov et al. (2013), which identifies phrases based on the occurrence frequency of unigrams and bigrams,

$$freq(w_i, w_j) = \frac{freq(w_i, w_j) - \delta}{freq(w_i) \times freq(w_j)} \quad (1)$$

where δ is a discounting coefficient that prevents too many phrases consisting of very infrequent words. We run 2-4 times over the corpora to get longer phrases containing more words. We empirically set δ as 10 in our experiment. We use the default frequency threshold (value=5) in the word2vec toolkit [2] to select bi-terms.

Given a sentence, we initialize the beam of each index with the current word, and sequentially add phrases into the beam if the new phrase is contained in the phrase table. At each index of a sentence, we rank the segmentation candidates by the inverted number of items within a segmentation, and save the top-ranked N segmentation candidates into the beam. An example of the generated segmentation candidates is given in Table 2.

Type	Sample
Sentence	that is not bad
Phrase Table	[is not], [not bad], [is not bad]
Segmentations	[that] [is not bad] [that] [is not] [bad] [that] [is] [not bad] [that] [is] [not] [bad]

Table 2: Example for segmentation candidate generation.

4.2 Segmentation Ranking Model

The objective of the segmentation ranking model is to assign a scalar to each segmentation candidate, which indicates the usefulness of the segmentation result for sentiment classification. In this subsection, we describe a log-linear model to calculate the segmentation score. To effectively train the segmentation ranking model, we devise a marginal log-likelihood as the optimization objective.

Given a segmentation candidate Ω_{ij} of the sentence s_i, we calculate the segmentation score for Ω_{ij} with a log-linear model, as given in Equation 2.

$$\phi_{ij} = exp(b + \sum_{k} sfe_{ijk} \cdot w_k) \qquad (2)$$

where ϕ_{ij} is the segmentation score of Ω_{ij}; sfe_{ijk} is the k-th segmentation feature of Ω_{ij}; w and b are the parameters of the segmentation ranking model.

During training, given a sentence s_i and its gold sentiment polarity pol_i^g, the optimization objective of the segmentation ranking model is to maximize the segmentation scores of the **hit candidates**, whose predicted sentiment polarity equals to the gold polarity of sentence pol_i^p. The loss function of the segmentation model is given in Equation 3.

$$loss = -\sum_{i=1}^{|T|} log(\frac{\sum_{j \in H_i} \phi_{ij}}{\sum_{j' \in A_i} \phi_{ij'}}) + \lambda ||w||_2^2 \qquad (3)$$

where T is the training data; A_i represents all the segmentation candidates of sentence s_i; H_i means the hit candidates of s_i; λ is the weight of the L2-norm regularization factor. We train the segmentation model with L-BFGS (Liu and Nocedal, 1989), running over the complete training data.

4.3 Feature

We design two kinds of features for sentence segmentation, namely the phrase-embedding feature and the segmentation-specific feature. The final feature representation of each segmentation is the concatenation of these two features. It is worth noting that, the phrase-embedding feature is used in both sentence segmentation and sentiment classification.

Segmentation-Specific Feature We empirically design four segmentation-specific features to reflect the information of each segmentation, as listed in Table 3.

Phrase-Embedding Feature We leverage phrase embedding to generate the features of segmentation candidates for both sentence segmentation and sentiment classification. The reason is that, in both tasks, the basic computational units of each segmentation candidate might be words or phrases of variable length. Under this scenario, phrase embedding is highly suitable as it is capable to represent phrases with different length into a consistent distributed vector space (Mikolov et al., 2013). For each phrase, phrase embedding is a dense, real-valued and continuous vector. After the phrase embedding is trained, the nearest neighbors in the embedding space are favored to have similar grammatical usages and semantic meanings. The effectiveness of phrase embedding has been verified for building large-scale sentiment lexicon (Tang et al., 2014a) and machine translation (Zhang et al., 2014).

We learn phrase embedding with Skip-Gram model (Mikolov et al., 2013), which is the state-of-

[2] Available at https://code.google.com/p/word2vec/

Feature	Feature Description
#unit	the number of basic computation units in the segmentation candidate
#unit / #word	the ratio of units' number in a candidate to the length of original sentence
#word − #unit	the difference between sentence length and the number of basic computational units
#unit > 2	the number of basic component units composed of more than two words

Table 3: Segmentation-specific features for segmentation ranking.

Feature	Feature Description
All-Caps	the number of words with all characters in upper case
Emoticon	the presence of positive (or negative) emoticons, whether the last unit is emoticon
Hashtag	the number of hashtag
Elongated units	the number of basic computational containing elongated words (with one character repeated more than two times), such as *gooood*
Sentiment lexicon	the number of sentiment words, the score of last sentiment words, the total sentiment score and the maximal sentiment score for each lexicon
Negation	the number of negations as individual units in a segmentation
Bag-of-Units	an extension of bag-of-word for a segmentation
Punctuation	the number of contiguous sequences of dot, question mark and exclamation mark.
Cluster	the presence of units from each of the 1,000 clusters from Twitter NLP tool (Gimpel et al., 2011)

Table 4: Classification-specific features for sentiment classification.

the-art phrase embedding learning algorithm. We compose the representation (or feature) of a segmentation candidate from the embedding of the basic computational units (words or phrases) it contains. In this paper, we explore min, max and $average$ convolution functions, which have been used as simple and effective methods for composition learning in vector-based semantics (Mitchell and Lapata, 2010; Collobert et al., 2011; Socher et al., 2013a; Shen et al., 2014; Tang et al., 2014b), to calculate the representation of a segmentation candidate. The final phrase-embedding feature is the concatenation of vectors derived from different convolutional functions, as given in Equation 4,

$$pf(seg) = [pf_{max}(seg), pf_{min}(seg), pf_{avg}(seg)] \quad (4)$$

where $pf(seg)$ is the representation of the given segmentation; $pf_x(seg)$ is the result of the convolutional function $x \in \{min, max, avg\}$. Each convolutional function $pf_x(\cdot)$ conducts the matrix-vector operation of x on the sequence represented by columns in the lookup table of phrase embedding. The output of $pf_x(\cdot)$ is calculated as

$$pf_x(seg) = \theta_x \langle L_{ph} \rangle_{seg} \quad (5)$$

where θ_x is the convolutional function of pf_x; $\langle L_{ph} \rangle_{seg}$ is the concatenated column vectors of the basic computational units in the segmentation; L_{ph} is the lookup table of phrase embedding.

5 Classification Model

For sentiment classification, we follow the supervised learning framework (Pang et al., 2002) and build the classifier from sentences with manually labelled sentiment polarity. We extend the state-of-the-art hand-crafted features in SemEval 2013 (Mohammad et al., 2013b), and design the classification-specific features for each segmentation. The detailed feature description is given in Table 4.

6 Experiment

In this section, we conduct experiments to evaluate the effectiveness of the joint model. We describe the experiment settings and the result analysis.

6.1 Dataset and Experiment Settings

We conduct sentiment classification of tweets on a benchmark Twitter sentiment classification dataset in SemEval 2013. We run 2-class (positive *vs* negative) classification as sentence segmentation has a great influence on the positive/negative polarity of tweets due to the *polarity inconsistency* between a phrase and its constitutes, such as ⟨*not bad*, *bad*⟩.

We leave 3-class classification (positive, negative, neutral) and fine-grained classification (very negative, negative, neutral, positive, very positive) in the future work.

	Positive	Negative	Total
Train	2,642	994	3,636
Dev	408	219	627
Test	1,570	601	2,171

Table 5: Statistics of the SemEval 2013 Twitter sentiment classification dataset (positive *vs* negative).

The statistics of our dataset crawled from SemEval 2013 are given in Table 5. The evaluation metric is the macro-F1 of sentiment classification. We train the joint model on the training set, tune parameters on the dev set and evaluate on the test set. We train the sentiment classifier with LibLinear (Fan et al., 2008) and utilize existing sentiment lexicons [3] to extract classification-specific features. We randomly crawl 100M tweets from February 1st, 2013 to April 30th, 2013 with Twitter API, and use them to learn the phrase embedding with Skip-Gram [4]. The vocabulary size of the phrase embedding is 926K, from unigram to 5-gram. The parameter -c in SVM is tuned on the dev-set in both baseline and our method. We run the L-BFGS for 50 iterations, and set the regularization factor λ as 0.003. The beam size N of the candidate generation model and the top-ranked segmentation number K are tuned on the dev-set.

6.2 Baseline Methods

We compare the proposed joint model with the following sentiment classification algorithms:

• *DistSuper*: We collect 10M balanced tweets selected by positive and negative emoticons [5] as training data, and build classifier using the LibLinear and ngram features (Go et al., 2009; Zhao et al., 2012).

• *SVM*: The n-gram features and Support Vector Machine are widely-used baseline methods to build sentiment classifiers (Pang et al., 2002). We use LibLinear to train the SVM classifier.

• *NBSVM*: NBSVM (Wang and Manning, 2012) trades-off between Naive Bayes and NB-features enhanced SVM. We use NBSVM-bi because it performs best on sentiment classification of reviews.

• *RAE*: Recursive Autoencoder (Socher et al., 2011) has been proven effective for sentiment classification by learning sentence representation. We train the RAE using the pre-trained phrase embedding learned from 100M tweets.

• *SentiStrength*: Thelwall et al. (2012) build a lexicon-based classifier which uses linguistic rules to detect the sentiment strength of tweets.

• *SSWE$_u$*: Tang et al. (2014b) propose to learn sentiment-specific word embedding (SSWE) from 10M tweets collected by emoticons. They apply SSWE as features for Twitter sentiment classification.

• *NRC*: NRC builds the state-of-the-art system in SemEval 2013 Twitter Sentiment Classification Track, incorporating diverse sentiment lexicons and hand-crafted features (Mohammad et al., 2013b). We re-implement this system because the codes are not publicly available. We do not directly report their results in the evaluation task, as our training and development sets are smaller than their dataset. In *NRC + PF*, We concatenate the NRC features and the phrase embeddings feature (*PF*), and build the sentiment classifier with LibLinear.

Except for *DistSuper*, other baseline methods are conducted in a supervised manner. We do not compare with *RNTN* (Socher et al., 2013b) because the tweets in our dataset do not have accurately parsed results. Another reason is that, due to the differences between domains, the performance of *RNTN* trained on movie reviews might be decreased if directly applied on the tweets (Xiao et al., 2013).

6.3 Results and Analysis

Table 6 shows the macro-F1 of the baseline systems as well as our joint model (**JSC**) on sentiment classification of tweets (positive *vs* negative).

As is shown in Table 6, distant supervision is relatively weak because the noisy-labeled tweets are treated as the gold standard, which decreases the performance of sentiment classifier. The result of bag-of-unigram feature (74.50%) is not satisfied as it losses the word order and does not well cap-

[3] In this work, we use *HL* (Hu and Liu, 2004), *MPQA* (Wilson et al., 2005), *NRC_Emotion_Lexicon* (Mohammad and Turney, 2012), *NRC_Hashtag_Lexicon* and *Sentiment140Lexicon* (Mohammad et al., 2013b).

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

[5] We use the emoticons selected by Hu et al. (2013). The positive emoticons are :) :) :-) :D =), and the negative emoticons are :(: (:-(.

Method	Macro-F1
DistSuper + unigram	61.74
DistSuper + 5-gram	63.92
SVM + unigram	74.50
SVM + 5-gram	74.97
Recursive Autoencoder	75.42
NBSVM	75.28
SentiStrength	73.23
$SSWE_u$	84.98
NRC (Top System in SemEval 2013)	84.73
NRC + PF	84.75
JSC	**85.51**

Table 6: Macro-F1 for positive *vs* negative classification of tweets.

ture the semantic meaning of phrases. The integration of high-order n-ngram (up to 5-gram) does not achieve significant improvement (+0.47%). The reason is that, if a sentence contains a bigram "*not bad*", they will use "*bad*" and "*not bad*" as parallel features, which confuses the sentiment classification model. *NBSVM* and *Recursive Autoencoder* perform comparatively and have a big gap in comparison with *JSC*. In *RAE*, the representation of a sentence is composed from the representation of words it contains. Accordingly, "*great*" in "*a great deal of*" also contributes to the final sentence representation via composition function. *JSC* automatically conducts sentence segmentation by considering the sentiment polarity of sentence, and utilize the phrasal information from the segmentations. Ideally, *JSC* regards phrases like "*not bad*" and "*a great deal of*" as basic computational units, and yields better classification performance. *JSC* (85.51%) performs slightly better than the state-of-the-art systems (*$SSWE_u$*, 84.98%; *NRC+PF*, 84.75%), which verifies its effectiveness.

6.4 Comparing Joint and Pipelined Models

We compare the proposed joint model with pipelined methods on Twitter sentiment classification with different feature sets. Figure 2 gives the experiment results. The tick *[A, B]* on x-axis means the use of *A* as segmentation feature and the use of *B* as classification feature. *PF* represents the phrase-embedding feature; *SF* and *CF* stand for the segmentation-specific feature and classification-specific feature, respectively. We use the bag-of-word segmentation result to build sentiment classier in *Pipeline 1*, and use the segmentation candidate with maximum phrase number in *Pipeline 2*.

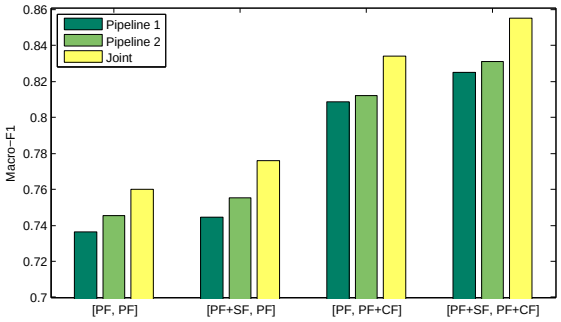


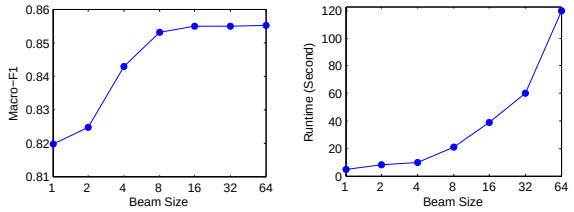
Figure 2: Macro-F1 for positive *vs* negative classification of tweets with joint and pipelined models.

From Figure 2, we find that the joint model consistently outperforms pipelined baseline methods in all feature settings. The reason is that the pipelined methods suffer from error propagation, since the errors from linguistic-driven and bag-of-word segmentations cannot be corrected by the sentiment classification model. Besides, traditional segmentors do not update the segmentation model with the sentiment information of text. Unlike pipelined methods, the joint model is capable to address these problems by optimizing the segmentation model with the classification results in a joint framework, which yields better performance on sentiment classification. We also find that *Pipeline 2* always outperforms *Pipeline 1*, which indicates the usefulness of phrase-based segmentation for sentiment classification.

6.5 Effect of the beam size N

We investigate the influence of beam size N, which is the maximum number of segmentation candidates of a sentence. In this part, we clamp the feature set as [PF+SF, PF+CF], and vary the beam size N in [1,2,4,8,16,32,64]. The experiment results of macro-F1 on the development set are illustrated in Figure 3 (a). The time cost of each training iteration is given in Figure 3 (b).

From Figure 3 (a), we can see that when larger beam size is considered, the classification performance is improved. When beam size is 1, the model stands for the greedy search with the bag-of-words segmentation. When the beam size is small, such as 2, beam search losses many phrasal information of sentences and thus the improvement is not significant. The performance remains steady when beam size is larger than 16. From



(a) Macro-F1 score for sentiment classification.

(b) Time cost (seconds) of each training iteration.

Figure 3: Sentiment classification of tweets with different beam size N.

Figure 3 (b), we can find that the runtime of each training iteration increases with larger beam size. It is intuitive as the joint model with larger beam considers more segmentation results, which increases the training time of the segmentation model. We set beam size as 16 after parameter learning.

6.6 Effect of the top-ranked segmentation number K

We investigate how the top-ranked segmentation number K affects the performance of sentiment classification. In this part, we set the feature as [PF+SF, PF+CF], and the beam size as 16. The results of macro-F1 on the development set are illustrated in Figure 4.

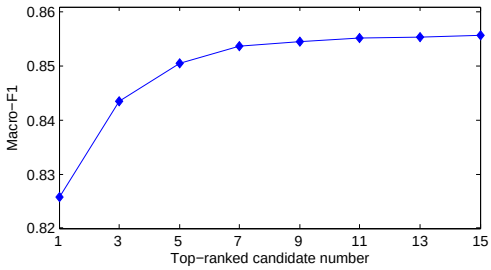


Figure 4: Sentiment classification of tweets with different top-ranked segmentation number K.

From Figure 4, we find that the classification performance increases with K being larger. The reason is that when a larger K is used, (1) at training time, the sentiment classifier is built by using more phrasal information from multiple segmentations, which benefits from the ensembles; (2) at test time, the joint model considers several top-ranked segmentations and get the final sentiment polarity through voting. The performance remains stable when K is larger than 7, as the phrasal information has been mostly covered.

7 Conclusion

In this paper, we develop a joint segmentation and classification framework (**JSC**) for sentiment analysis. Unlike existing sentiment classification algorithms that build sentiment classifier based on the segmentation results from bag-of-words or separate segmentors, the proposed joint model simultaneously conducts sentence segmentation and sentiment classification. We introduce a marginal log-likelihood function to optimize the segmentation model, and effectively train the joint model from sentences annotated only with sentiment polarity, without segmentation annotations of sentences. The effectiveness of the joint model has been verified by applying it on the benchmark dataset of Twitter sentiment classification in SemEval 2013. Results show that, the joint model performs comparably with state-of-the-art methods, and outperforms pipelined methods in various settings. In the future, we plan to apply the joint model on other domains, such as movie/product reviews.

Acknowledgements

We thank Nan Yang, Yajuan Duan, Yaming Sun and Meishan Zhang for their helpful discussions. We thank the anonymous reviewers for their insightful comments and feedbacks on this work. This research was partly supported by National Natural Science Foundation of China (No.61133012, No.61273321, No.61300113). The contact author of this paper, according to the meaning given to this role by Harbin Institute of Technology, is Bing Qin.

References

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa.

2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 231–240.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 49–54.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 42–47.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. *Proceedings of International Conference on Machine Learning*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

G.E. Hinton and R.R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Ming Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. 2013. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the International World Wide Web Conference*, pages 607–618.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. *The Proceeding of Annual Meeting of the Association for Computational Linguistics*.

Nicola Jones. 2014. Computer science: The learning machines. *Nature*, 505(7482):146.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A sentence model based on convolutional neural networks. In *Procedding of the 52th Annual Meeting of Association for Computational Linguistics.*

Hyun Duk Kim and ChengXiang Zhai. 2009. Generating comparative summaries of contradictory opinions in text. In *Proceedings of CIKM 2009*. ACM.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *The International AAAI Conference on Weblogs and Social Media*.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Annual Meeting of the Association for Computational Linguistics*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of international conference on Machine learning*. ACM.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Proceedings of International Conference on Machine Learning*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Conference on Neural Information Processing Systems*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654.

Saif M Mohammad and Peter D Turney. 2012. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*.

Saif M Mohammad, Bonnie J Dorr, Graeme Hirst, and Peter D Turney. 2013a. Computing lexical contrast. *Computational Linguistics*, 39(3):555–590.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013b. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *Proceedings of the International Workshop on Semantic Evaluation*.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of Language Resources and Evaluation Conference*, volume 2010.

Georgios Paltoglou and Mike Thelwall. 2010. A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1386–1395.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–86.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 373–374.

Richard Socher, J. Pennington, E.H. Huang, A.Y. Ng, and C.D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Y Ng. 2013a. Reasoning with neural tensor networks for knowledge base completion. *The Conference on Neural Information Processing Systems*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014a. Building large-scale twitter-specific sentiment lexicon : A representation learning approach. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 172–182.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, 63(1):163–173.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 417–424.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 90–94.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 347–354.

Min Xiao, Feipeng Zhao, and Yuhong Guo. 2013. Learning latent word representations for domain adaptation using supervised word clustering. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 152–162, October.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 172–182.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 111–121.

Jichang Zhao, Li Dong, Junjie Wu, and Ke Xu. 2012. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*.