A Joint Segmentation and Classification Framework for Sentiment Analysis

Duyu Tang\textsuperscript{1}, Furu Wei\textsuperscript{2}, Bing Qin\textsuperscript{1}, Li Dong\textsuperscript{3}, Ting Liu\textsuperscript{1}, Ming Zhou\textsuperscript{2}

\textsuperscript{1} Harbin Institute of Technology
\textsuperscript{2} Microsoft Research
\textsuperscript{3} Beihang University
Sentence-level sentiment classification

- **Input:** A sentence (e.g. tweet)
- **Output:** Sentiment polarity of the sentence
  - Positive / Negative
Existing Methods: Pipelined

- Training Data
- Segmentation Algorithm
- Segmentation Results
- Classification Algorithm
- Polarity Label
Existing Methods: Pipelined

- Training Data
- Segmentation Algorithm
- Segmentation Results
- Classification Algorithm
- Polarity Label

that is not bad

\[
\begin{align*}
&that \\
is \\
not \\
bad
\end{align*}
\]

Positive (+1)
Existing Methods: Pipelined

Polarity inconsistency between a phrase and the words it contains.

\[
\begin{align*}
\text{not bad} & \quad \leftrightarrow \quad \text{bad} \\
\text{a great deal of} & \quad \leftrightarrow \quad \text{great}
\end{align*}
\]
A Joint Model

Training Data → Segmentation Results → Classification Algorithm

Segmentation Algorithm → Polarity Label
A Joint Model

Segmentation => Classification

Segmentation result is the input for classification.

Classification => Segmentation

Classification answer (right or wrong) can indicate the usefulness of a segmentation.
The Inference Process

*that is not bad*
The Inference Process

The candidate generation model
The Inference Process

Input

Segmentations

Rank

that is not bad

that is not bad

that is not bad

that is not bad

CG

SEG

The candidate generation model

The segmentation ranking model

that is not bad

0.6

0.4

2.3

1.6
The Inference Process

- **CG**: The candidate generation model
- **SEG**: The segmentation ranking model

Input: that is not bad

Segmentations:
- that is not bad
- that is not bad
- that is not bad
- that is not bad

Rank:
- Top K: 2.3, 1.6
- CG: 0.6, 0.4
The Inference Process

The candidate generation model

The sentiment classification model

The segmentation ranking model
The Inference Process

Input

Segmentations

Rank

Polarity

Vote

that is not bad

The candidate generation model

The sentiment classification model

The segmentation ranking model
The Training Process

*that is not bad*

Polarity: +1
The Training Process

The candidate generation model

The use of a model

that is not bad

Polarity: +1
The Training Process

that is not bad

Polarity: +1

Input

Segmentations

Polarity

CG

The candidate generation model

SC

The sentiment classification model

The use of a model
The Training Process

Polarity: +1

Input: *that is not bad*

Segmentations:
- *that is not bad*  
  Polarity: -1  
  Update: <+1,-1> NO
- *that is not bad*  
  Polarity: -1  
  Update: <+1,-1> NO
- *that is not bad*  
  Polarity: +1  
  Update: <+1,+1> YES
- *that is not bad*  
  Polarity: +1  
  Update: <+1,+1> YES

The candidate generation model
The sentiment classification model
The segmentation ranking model

The use of a model
The update of a model
The Training Process

that is not bad

Polarity: +1

The candidate generation model
The sentiment classification model
The segmentation ranking model

CG
The use of a model

SC
The update of a model

SEG

that is not bad -1 <+1,-1> NO 0.6
that is not bad -1 <+1,-1> NO 0.4
that is not bad +1 <+1,+1> YES 2.3
that is not bad +1 <+1,+1> YES 1.6
The Training Process

**Input**

1. that is not bad
2. that is not bad
3. that is not bad
4. that is not bad

**Segmentations**

1. that is not bad
2. that is not bad
3. that is not bad
4. that is not bad

**Polarity**: +1

The candidate generation model

The sentiment classification model

The segmentation ranking model

**CG**

The use of a model

**SC**

The update of a model

**SEG**

Top K
• Candidate Generation Model

• Segmentation Ranking Model

• Sentiment Classification Model
Candidate Generation Model

• A Beam-Search Approach
  – **Step 1**: Learn a phrase table from a large corpora with **word2vec**
  – **Step 2**: Get the possible segmentations with beam-search.
Candidate Generation Model

• A Beam-Search Approach
  – **Step 1**: Learn a **phrase table** from a large corpora with **word2vec**
  – Intuition
    • Identify phrases based on the occurrence frequency of unigrams and bigrams
    \[ freq(w_i, w_j) = \frac{freq(w_i, w_j) - \delta}{freq(w_i) \times freq(w_j)} \]
  – Run 2-4 times over the corpora to get longer phrases containing more words
Candidate Generation Model

• A Beam-Search Approach
  – **Step 2**: Beam-Search

<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam</td>
</tr>
</tbody>
</table>

Phrase Table

- is not bad
- not bad

Sentence

<table>
<thead>
<tr>
<th>that</th>
<th>is</th>
<th>not</th>
<th>bad</th>
</tr>
</thead>
</table>

Phrase Table

- that
Candidate Generation Model

• A Beam-Search Approach
  – Step 2: Beam-Search

The bigram “that it” is NOT contained in the phrase table.
Candidate Generation Model

• A Beam-Search Approach
  – **Step 2:** Beam-Search

The bigram "is not" is contained in the phrase table.
Candidate Generation Model

• A Beam-Search Approach
  – Step 2: Beam-Search

The bigram “not bad” is contained in the phrase table.
Candidate Generation Model

• A Beam-Search Approach
  – **Step 2**: Beam-Search

The trigram “is not bad” is contained in the phrase table.
• Candidate Generation Model

• Segmentation Ranking Model

• Sentiment Classification Model
Segmentation Ranking Model

\[ y = f(x) \]

Segmentation

Score

\[ y = f(x) \]

that is not bad

that is not bad

that is not bad

0.6

2.3

1.6

Sentence

Segmentation

Segmentation Score
A Log-Linear Function

\[ y = f(x) \]

\[ y = \exp(b + \sum_i W_i \cdot F_i) \]

Segmentation \hspace{2cm} Score

Weight \hspace{2cm} Feature
# Segmentation Features

<table>
<thead>
<tr>
<th>Feature 1</th>
<th>Feature Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#unit</td>
<td>the number of basic computation units in the segmentation candidate</td>
</tr>
<tr>
<td>#unit/#word</td>
<td>the ratio of units’ number in a candidate to the length of original sentence</td>
</tr>
<tr>
<td>#word − #unit</td>
<td>the difference between sentence length and the number of basic computational units</td>
</tr>
<tr>
<td>#unit &gt; 2</td>
<td>the number of basic computation units composed of more than two words</td>
</tr>
</tbody>
</table>

| Feature 2       | Phrase embedding features learned from SkipGram.                                    |
Segmentation Features

### Feature 1

<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#unit</td>
<td>the number of basic computation units in the segmentation candidate</td>
</tr>
<tr>
<td>#unit/#word</td>
<td>the ratio of units’ number in a candidate to the length of original sentence</td>
</tr>
<tr>
<td>#word – #unit</td>
<td>the difference between sentence length and the number of basic computational units</td>
</tr>
<tr>
<td>#unit &gt; 2</td>
<td>the number of basic computation units composed of more than two words</td>
</tr>
</tbody>
</table>

### Feature 2

Phrase embedding features learned from SkipGram.

---

<Diagram>

- **Embedding Layer**
  - Inputs: "the", "weather", "is", "not bad", :) .
  - Layers: $f$, $f'$
  - Operations: average, max, min, concatenate

---

32
Training Objective

\[
x \xrightarrow{\text{Segmentation}} y = f(x) \xrightarrow{\text{Score}} y
\]

that is not bad

Positive (+1)

Sentence

Segmentation
Training Objective

\[ y = f(x) \]

Segmentation

Score

\[ y = f(x) \]

0.6

2.3

1.6

Sentence

Segmentation

Segmentation Score

Positive (+1)

that is not bad

that | is | not | bad

that | is | not bad

that is | not bad
Training Objective

\[ y = f(x) \]

Segmentation: \( x \rightarrow y \)

Score: \( y = f(x) \)

Sentence: "that is not bad"

Segmentation Score: 0.6, 2.3, 1.6

Polarity: -1, +1, +1

Positive (+1)
Training Objective

\[ y = f(x) \]

Segmentation

Score

\[ y = f(x) \]

that is not bad

Positive (+1)

Sentence

Segmentation

Segmentation Score

Polarity

Hit candidates

Maximize the scores of the hit candidates.
Training Objective

\[ y = \exp(b + \sum_i W_i \cdot F_i) \]

\[ \text{loss} = - \sum_{i=1}^{|T|} \log \left( \frac{\sum_{j \in H_i} y_{ij}}{\sum_{j' \in A_i} y_{ij'}} \right) + \lambda \sum_k W_k^2 \]

Maximize the scores of the hit candidates.
• Candidate Generation Model

• Segmentation Ranking Model

• Sentiment Classification Model
Sentiment Classification Model

- A supervised learning framework

Training Data → Feature Representation → Learning Algorithm → Sentiment Classifier

Massive Tweets → Embedding Learning → Embedding Feature → NRC-Canada Feature

- Dimension 1
- Dimension 2
- All-cap emoticon
- Elongated

NRC-Canada Feature: 1 2 \ldots N N+1 N+2 \ldots N+K
## Classification Feature

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

| Feature 2                  | Phrase embedding features learned from SkipGram.                                      |
Experiment Results

• Polarity Classification of Tweets
  – Positive/Negative
  – Benchmark dataset from SemEval 2013

<table>
<thead>
<tr>
<th></th>
<th>Positive</th>
<th>Negative</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>2,642</td>
<td>994</td>
<td>3,636</td>
</tr>
<tr>
<td>Dev</td>
<td>408</td>
<td>219</td>
<td>627</td>
</tr>
<tr>
<td>Test</td>
<td>1,570</td>
<td>601</td>
<td>2,171</td>
</tr>
</tbody>
</table>
Results

• Comparison with classification methods
Results

• Comparison with pipelined methods

**Pipeline 1**: Bag-of-word segmentation

**Pipeline 2**: Segmentation with maximum phrases.
Summary

• We develop a **joint** model to learn sentiment-specific sentence segmentor for sentiment classification

• Our method yields **comparable** performance with the state-of-the-art methods on Twitter sentiment classification.
Thanks