Thumbs up?
Sentiment Classification using
Machine Learning Techniques

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?
Sentiment Classification using Machine Learning Techniques. EMNLP-2002,
79—86.

Kalaiselvan Panneerselvam
Outline

• What is Sentiment Analysis
• Sentiment Classification in Movie Reviews
• Baseline Classifier
• Supervised Learning Process
• Framework
• Feature Extraction
• Classifiers
• Results
• References
Terms

• Sentiment
  • A thought, view, or attitude, especially one based mainly on emotion instead of reason

• Sentiment Analysis
  • aka opinion mining
  • use of natural language processing (NLP) and computational techniques to automate the extraction or classification of sentiment from typically unstructured text
Sentiment Analysis

What is it used for?

• Natural language and text processing to identify and extract subjective information

• Classifying the *polarity* of a given text as *positive*, *negative* or *neutral*

• In general: to discover how people *feel* about a particular topic
Sentiment Analysis

who is it used by?

• Consumer information
  • Product reviews

• Marketing
  • Consumer attitudes
  • Trends

• Politics
  • Politicians want to know voters’ views
  • Voters want to know politicians’ stances and who else supports them

• Social
  • Find like-minded individuals or communities
HP Officejet 6500A E710N Multifunction Printer

Average rating ★★★★★ (144)  Most mentioned  Show reviews by source
★★★★★ (55) Performance (57) Best Buy (140)
★★★★ (54) Ease of Use (43) CNET (5)
★★★★ (10) Print Speed (39) Amazon.com (3)
★★★ (6) Connectivity (31)
★★ (23)
★ (6)
★ (0)

$121.53 - $242.39 (14 stores)

Compare
Sentiment Classification in Movie Reviews


• Polarity detection:
  • Is an IMDB movie review positive or negative?

• Data: Polarity Data 2.0:
  • http://www.cs.cornell.edu/people/pabo/movie-review-data
  • Use reviews with star or numerical value as training and test data.
when _star wars_ came out some twenty years ago, the image of traveling throughout the stars has become a commonplace image. […] when han solo goes light speed, the stars change to bright lines, going towards the viewer in lines that converge at an invisible point. cool.

_october sky_ offers a much simpler image—that of a single white dot, traveling horizontally across the night sky. […]

“snake eyes” is the most aggravating kind of movie: the kind that shows so much potential then becomes unbelievably disappointing. it’s not just because this is a brian depalma film, and since he’s a great director and one who’s films are always greeted with at least some fanfare. and it’s not even because this was a film starring nicolas cage and since he gives a brauvara performance, this film is hardly worth his talents.
The Data

- **Internet Movie Database (IMDB) archive**

- **Limited data to:**
  - Reviews with author rating
  - Positive and negative reviews (no neutral)
  - 19 positive, 19 negative reviews per author

- **Interim Dataset:**
  - 752 negative reviews
  - 1301 positive reviews
  - 144 reviewers represented

- **Final Dataset:** 700 positive, 700 negative (uniform distribution)
Prior Work

• Prior classification based on:
  • source/source style
  • genre
  • knowledge-based
  • semantic orientation using text categorization
Positive or negative movie review?

• unbelievably disappointing
• Full of zany characters and richly applied satire, and some great plot twists
• this is the greatest screwball comedy ever filmed
• It was pathetic. The worst part about it was the boxing scenes.
## Baseline (Human Classifier)

<table>
<thead>
<tr>
<th></th>
<th>Positive List</th>
<th>Negative List</th>
<th>Accuracy</th>
<th>Ties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Human 1</strong></td>
<td>dazzling, brilliant, phenomenal, excellent, fantastic</td>
<td>suck, terrible, awful, unwatchable, hideous</td>
<td>58%</td>
<td>75%</td>
</tr>
<tr>
<td><strong>Human 2</strong></td>
<td>gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting</td>
<td>bad, clichéd, sucks, boring, stupid, slow</td>
<td>64%</td>
<td>39%</td>
</tr>
<tr>
<td><strong>Human 3 + stats</strong></td>
<td>love, wonderful, best, great, superb, still, beautiful</td>
<td>bad, worst, stupid, waste, boring, ?, !</td>
<td>69%</td>
<td>16%</td>
</tr>
</tbody>
</table>

- **Crafted word lists** using independent CS grad students
- **Positive vs. negative word count**
- **Frequency counts** (including test data)
- **Hand-picked words**

*Tie rates - Percentage of documents where the two sentiments were rated equally likely.*
Thumbs up? Sentiment Classification using Machine Learning Techniques

• From above experiment, it is proved worthwhile to explore corpus-based techniques, rather than relying on prior intuitions, to select good indicator features and to perform sentiment classification in general.
Supervised Learning Process

Supervised Learning Process: Two Steps

- **Learning (training)**
  - Learn a model using the *training data*

- **Testing**
  - Test the model using *unseen test data* to assess the model accuracy
  - *Accuracy* = \( \frac{\text{Number of correct classifications}}{\text{Total number of test cases}} \)

---

Step 1: training

Step 2: testing
Framework

Training Model

- Movie Reviews
- Develop Features
- NB
- ME
- SVM
- Evaluate Results
- Extract Insights
Framework (continued…)

- **Tokenization**
  - Treated punctuation as separate lexical items
  - No stemming or stoplists were used.

- **Feature Extraction**
  - Unigram, Bigram, POS, and combinations of features

- **Classification using different classifiers**
  - Naïve Bayes
  - MaxEnt
  - Support Vector Machines
Sentiment Tokenization Issues

• Deal with HTML and XML markup
• Twitter mark-up (names, hash tags)
• Capitalization (preserve for words in all caps)
• Phone numbers, dates and Emoticons
• Commonly used tokenizer:
  • Christopher Potts sentiment tokenizer
  • Brendan O’Connor twitter tokenizer
Extracting Features for Sentiment Classification

- How to handle negation
  - I didn’t like this movie
  - vs
  - I really like this movie

- Which words to use?
  - Only adjectives
  - All words
    - All words turns out to work better, at least on this data
Extracting Features (Continued...)

• Features
  • Unigrams: A single word.
  • Feature frequency: frequency of a feature appears
  • Feature presence: 1 only when a feature appears
  • Bigrams: Two continues word.
  • Parts of Speech: Tag the word with its POS.
  • Adjectives: Only use adjectives in the text.
  • Position: The position of a word in the text. In the first quarter, last quarter or the middle half.
Classifiers

Approaches

1. Naïve Bayes Classifier
2. Max Entropy
3. Support vector machine
Naïve Bayes Classifiers

- **Naïve Bayes Classifier**
  - Simple classification of words based on ‘Bayes theorem’.
  - It is a ‘Bag of words’ (text represented as collection of it’s words, discarding grammar and order of words but keeping multiplicity) approach for subjective analysis of a content.
  - **Application**: Sentiment detection, Email spam detection, Document categorization etc.
  - Superior in terms of CPU and Memory utilization as shown by Huang, J. (2003).
Naïve Bayes Classifiers (Continued...)

- **Probabilistic Analysis of Naïve Bayes**

For a document $d$ and class $c$, By Bayes theorem

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Naïve Bayes Classifier will be - :

$$c^* = \arg \max_c P(c \mid d)$$
Naïve Bayes Classifiers (Continued...)

Naïve Bayes:

\[
P(c \mid d) = \frac{P(c)P(d \mid c)}{P(d)},
\]

given document \(d\), the class \(c^* = \arg \max c P(c \mid d)\).

Assume all features are conditionally independent

\[
P_{NB}(c \mid d) := \frac{P(c)\left(\prod_{i=1}^{m} P(f_i \mid c)^{n_i(d)}\right)}{P(d)}.
\]

\(n_i(d)\) is the number of times \(f_i\) occurs in document \(d\).

\[
P(Sentiment \mid Sentence) = \frac{P(Sentiment \mid Sentence)P(Sentiment)}{P(Sentence)}
\]

\[
P(Word \mid Sentiment) = \frac{\text{Number of words occurring in class} + 1}{\text{Number of words belonging to class} + \text{Total number of words}}
\]

Course 3111: Seminar: Data Analytics I
Example

POSITIVE POLARITY (GOOD)
- a mesmerizing cinematic poem from the first frame to the last.
- a well-put-together piece of urban satire.
- one can't deny its seriousness and quality.
- hard to resist.
- a naturally funny film, home movie makes you crave Chris Smith's next movie.
- a true-blue delight.
- a fun ride.
- a surprisingly funny movie.
- the script is smart and dark - hallelujah for small favors.
- a flick about our infantilized culture that isn't entirely infantile.

NEGATIVE POLARITY (BAD)
- unfortunately the story and the actors are served with a hack script.
- too slow for a younger crowd, too shallow for an older one.
- terminally brain dead production.
- one lousy movie.
- this movie... doesn't deserve the energy it takes to describe how bad it is.
- a cleverly crafted but ultimately hollow mockumentary.
- it's an 88-minute highlight reel that's 86 minutes too long.
- the whole affair is as predictable as can be.
Example(Continued...)

“it’s rather like a lifetime special -- pleasant, sweet and forgettable. “

Good: 46
Bad: 22
Goodness: 46/(46+22) = 0.68
Badness: 22/(46+22) = 0.32

Good: 506
Bad: 507
Goodness: 506/(506+507) = 0.5
Badness: 507/(506+507) = 0.5

“it’s rather like a lifetime special -- pleasant, sweet and forgettable. “

Good: 15
Bad: 6
Goodness: 15/(6+15) = 0.71
Badness: 6/(6+15) = 0.29

Good: 46
Bad: 22
Goodness: 46/(46+22) = 0.68
Badness: 22/(46+22) = 0.32
Example (Continued...)

```
"it's rather like a lifetime special -- pleasant, sweet and forgettable."

<table>
<thead>
<tr>
<th></th>
<th>#GOOD</th>
<th>#BAD</th>
<th>GOODNESS</th>
<th>BADNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>it's</td>
<td>506</td>
<td>507</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>rather</td>
<td>42</td>
<td>63</td>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>like</td>
<td>242</td>
<td>396</td>
<td>0.61</td>
<td>0.39</td>
</tr>
<tr>
<td>a</td>
<td>3446</td>
<td>3112</td>
<td>0.53</td>
<td>0.47</td>
</tr>
<tr>
<td>lifetime</td>
<td>3</td>
<td>5</td>
<td>0.38</td>
<td>0.62</td>
</tr>
<tr>
<td>special</td>
<td>29</td>
<td>40</td>
<td>0.42</td>
<td>0.58</td>
</tr>
<tr>
<td>pleasant</td>
<td>15</td>
<td>6</td>
<td>0.71</td>
<td>0.29</td>
</tr>
<tr>
<td>sweet</td>
<td>46</td>
<td>22</td>
<td>0.68</td>
<td>0.32</td>
</tr>
<tr>
<td>and</td>
<td>3198</td>
<td>2371</td>
<td>0.57</td>
<td>0.43</td>
</tr>
<tr>
<td>forgettable</td>
<td>10</td>
<td>14</td>
<td>0.42</td>
<td>0.58</td>
</tr>
</tbody>
</table>

"BAG OF WORDS" model

SUM: 5.22 4.8

So we should classify this as a POSITIVE review!
```
Maximum Entropy

- Maximum Entropy is a technique for learning probability distributions from data

- “Don’t assume anything about your probability distribution other than what you have observed.”

- Always choose the most uniform distribution subject to the observed constraints.
Maximum Entropy

- Maximum Entropy:

\[
P_{ME}(c \mid d) := \frac{1}{Z(d)} \exp \left( \sum_{i} \lambda_{i,c} F_{i,c}(d, c) \right),
\]

\[
F_{i,c}(d, c') := \begin{cases} 
1, & n_i(d) > 0 \text{ and } c' = c \\
0, & \text{otherwise}
\end{cases}
\]

\(Z(d)\) is a normalization function. The \(\lambda_{i,c}\)'s are feature-weight parameters. Larger \(\lambda_{i,c}\)'s mean \(f_i\) is considered a strong indicator for class \(c\).
Support Vector Machine

Find a hyperplane that makes the margin between two categories.

SVM creates a hyperplanes or a set of hyperplanes in infinite dimension space. The SVM score $z_j$ of a document is mathematically given as follows:

$$z_j = w_1x_{j1} + w_2x_{j2} + \ldots... + w_dx_{jd} + b$$

i.e. $z_j = x_j^Tw + b$

where,
- $x_i$ is a $p$-dimensional real vector.
- $w$ is vector that contains the weights and is given as

$$\overline{w} = \sum_j \alpha_j c_j \vec{d}_j, \quad \alpha_j \geq 0, \ c_j = \{1, -1\}$$

$b$ is a constant
Scenario 1

**Identify the right hyper-plane (Scenario-1):**

- Here, we have three hyper-planes (A, B and C). Now identify the right hyper-plane to classify star and circle.
- You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.
Scenario 2

Identify the right hyper-plane (Scenario-2):

Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?

Here, maximizing the distances between nearest data point (either class) and hyperplane will help us to decide the right hyper-plane. This distance is called as Margin.

Above, you can see that the margin for hyperplane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyperplane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of mis-classification.
Results

• Results for different feature:

<table>
<thead>
<tr>
<th>Features</th>
<th># of features</th>
<th>frequency or presence?</th>
<th>NB</th>
<th>ME</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) unigrams</td>
<td>16165</td>
<td>freq.</td>
<td>78.7</td>
<td>N/A</td>
<td>72.8</td>
</tr>
<tr>
<td>(2) unigrams</td>
<td></td>
<td>pres.</td>
<td>81.0</td>
<td>80.4</td>
<td>82.9</td>
</tr>
<tr>
<td>(3) unigrams+bigrams</td>
<td>32330</td>
<td>pres.</td>
<td>80.6</td>
<td>80.8</td>
<td>82.7</td>
</tr>
<tr>
<td>(4) bigrams</td>
<td>16165</td>
<td>pres.</td>
<td>77.3</td>
<td>77.4</td>
<td>77.1</td>
</tr>
<tr>
<td>(5) unigrams+POS</td>
<td>16695</td>
<td>pres.</td>
<td>81.5</td>
<td>80.4</td>
<td>81.9</td>
</tr>
<tr>
<td>(6) adjectives</td>
<td>2633</td>
<td>pres.</td>
<td>77.0</td>
<td>77.7</td>
<td>75.1</td>
</tr>
<tr>
<td>(7) top 2633 unigrams</td>
<td>2633</td>
<td>pres.</td>
<td>80.3</td>
<td>81.0</td>
<td>81.4</td>
</tr>
<tr>
<td>(8) unigrams+position</td>
<td>22430</td>
<td>pres.</td>
<td>81.0</td>
<td>80.1</td>
<td>81.6</td>
</tr>
</tbody>
</table>

• Unigrams works better than baseline
• Presence is better than frequency
• Bigram feature does not improve performance
• Adjectives are poor
• POS improve slight for NB and ME, but decline for SVM
• Position also does not help
Insights

• SVM is found to be more accurate.
• Not comparable to topic-based categorization models.
• Simple unigram presence the best.
• Presence > Frequency, not like topic-based.
• Uncovered “thwarted expectations” narrative.
  • “Okay, I’m really ashamed of it, but I enjoyed it. I mean, I admit it’s a really awful movie.”
References

Thank you
Twitter as a corpus for Sentiment Analysis and Opinion Mining

Alexander Pak, Patrick Paroubek
Université de Paris-Sud, LREc, 2010
Abstract

• Millions of users share opinions on microblogging sites
• Rich data source for opinion mining and sentiment analysis
• Focus on twitter corpus collection for sentiment analysis
• Corpus is used to build a sentiment classifier
• Claims that the proposed techniques work better than previous methods
Introduction

- Microblogging has become very popular nowadays
- People generally write about their life, share opinions and discuss current issues
- Users post about products & services, express political and religious views
- Twitter has enormous number of text posts and it grows everyday
- Audience varies from regular users, to celebs to politicians and even president
- This can be used for marketing or social studies
- Authors performed the following steps:
  - Collected 300,000 tweets, divided them into 3 sets: Positive, Negative and Neutral
  - Performed linguistic analysis of collected corpus
  - Built a sentiment classifier
  - Experimental Evaluation
Corpus Collection

- Used **twitter API** to collect **positive, negative** and **objective** sentiments

- For positive and negative sentiments, used **emoticon** approach:
  - Happy emoticons: “:-)”, “:)”, “=)”, “:D” etc.
  - Sad emoticons: “-(”, “:(”, “(”, “(”, “;)” etc.

- For **objective** posts, tweets pulled from 44 popular **news outlets** such as ‘Washington Post’ and ‘New York Times’

- **Assumption**: Due to short character limit, it was assumed that emoticons represent the sentiment for the entire tweet

- English language was used for research, however, the method can be easily adapted to other languages
Corpus Analysis

- Word frequency distribution follows **Zipf’s law**
- This confirms a proper characteristic of the corpus
- Use **TreeTagger** (Schmid 1994) for POS tagging
- Compared tag distributions between sets of texts
- Pairwise comparison of each tag done by calculating:
  \[
  p_{1,2}^T = \frac{N_1^T - N_2^T}{N_1^T + N_2^T}
  \]
- Where \(N_i\) is the number of occurrences of tag \(T\) in set \(i\)
- Compared two sets of data:
  i. Subjective vs. Objective
  ii. Negative vs. Positive

![Figure 1: The distribution of the word frequencies follows Zipf’s law](image)
TreeTagger

- TreeTagger is a tool for annotating text with part-of-speech and lemma information

<table>
<thead>
<tr>
<th>word</th>
<th>pos</th>
<th>lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>DT</td>
<td>the</td>
</tr>
<tr>
<td>TreeTagger</td>
<td>NP</td>
<td>TreeTagger</td>
</tr>
<tr>
<td>is</td>
<td>VBZ</td>
<td>be</td>
</tr>
<tr>
<td>easy</td>
<td>JJ</td>
<td>easy</td>
</tr>
<tr>
<td>to</td>
<td>TO</td>
<td>to</td>
</tr>
<tr>
<td>use</td>
<td>VB</td>
<td>use</td>
</tr>
<tr>
<td>.</td>
<td>SENT</td>
<td>.</td>
</tr>
</tbody>
</table>

Figure 1: A sample decision tree
### Corpus Analysis

<table>
<thead>
<tr>
<th>Objective</th>
<th>Subjective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains more common and proper nouns (NPS, NP, NNS)</td>
<td>Contains more personal pronouns (PP, PP$)</td>
</tr>
<tr>
<td>Verbs are usually in 3rd person (VBZ)</td>
<td>Verbs are in 1st of 2nd person (VBP)</td>
</tr>
<tr>
<td>past participle (VBN) used</td>
<td>Use simple past tense (VBD)</td>
</tr>
<tr>
<td>Comparative adjectives (JJR) used for stating facts</td>
<td>Superlative adjectives (JJS) used for expressing emotions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superlative adverbs and possessive endings (POS)</td>
<td>Verbs in past tense (VBN, VBD)</td>
</tr>
</tbody>
</table>

---

**Figure 2:** $P^T$ values for objective vs. subjective

**Figure 3:** $P^T$ values for positive vs. negative
Training Classifier

• Use a combination of n-gram approach and POS tagging for sentiment analysis

• Experimented with unigrams, bigrams and trigrams to find best settings for microblogging data

• **Data Preparation:**
  i. **Filtration:** Remove URLs, user names and special words like ‘RT’
  ii. **Tokenization:** Split text to create bag of words
  iii. **Remove Stopwords:** Remove articles such as ‘a’, ‘an’, ‘the’ from bag of words
  iv. **Construct n-grams:** Ensure that negation is attached to the word and considered as a single word

• Used naïve bayes classifier

\[
P(s|M) = \frac{P(s) \cdot P(M|s)}{P(M)} \quad \Rightarrow \quad P(s|M) = \frac{P(M|s)}{P(M)} \quad \Rightarrow \quad P(s|M) \sim P(M|s)
\]

where \( s \) is sentiment, \( M \) is a message
Training Classifier

• We train 2 bayes classifiers: One with n-grams as features, another with POS distribution

• **Assumption**: POS tags are conditionally independent of n-grams

\[ P(s|M) \sim P(G|s) \cdot P(T|S) \]

• Where G is the set of N-grams representing a message and T is the set of POS-tags

• **Assumption**: n-grams and POS-tags are also conditionally independent

• For all

\[ P(s|M) \sim \prod_{g \in G} P(g|s) \cdot \prod_{t \in G} P(t|s) \]

• Finally we take the log likelihood of each sentiment

\[ L(s|M) = \sum_{g \in G} \log(P(g|s)) + \sum_{t \in G} \log(P(t|s)) \]
Increasing Accuracy

• To increase accuracy, we calculate **entropy** of a probability distribution

\[
entropy(g) = H(p(S | g)) = - \sum_{i=1}^{N} p(S_i | g) \log p(S_i | g)
\]

• **High entropy** indicates that the distribution of **sentiments** in different sentiment datasets is close to **uniform**

• These n-grams can be discarded as they do not contribute much to the classification

• We introduce another term, **salience**

\[
salience(g) = \frac{1}{N} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} 1 - \frac{\min(P(g|s_i), P(g|s_j))}{\max(P(g|s_i), P(g|s_j))}
\]

• Salience takes a value between 0 and 1. Low value n-grams should be discarded

• The final equation remains the same, except the fact that we discard n-grams with entropy higher than \( \theta \) or salience lower than \( \theta \)
Model Validation

- Classifier was tested on a set of real hand-annotated twitter posts

\[
\text{accuracy} = \frac{N(\text{correct classifications})}{N(\text{all classifications})}
\]

\[
\text{decision} = \frac{N(\text{retrieved documents})}{N(\text{all documents})}
\]

<table>
<thead>
<tr>
<th>Sentiment</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>108</td>
</tr>
<tr>
<td>Negative</td>
<td>75</td>
</tr>
<tr>
<td>Neutral</td>
<td>33</td>
</tr>
<tr>
<td>Total</td>
<td>216</td>
</tr>
</tbody>
</table>
Results

- Best results are achieved when using bi-grams
- Bigrams have good balance between coverage, and ability to capture sentiment patterns
- The model has high accuracy, however lower decision value
- Classifier is optimal for a sentiment search engine
- F-measure is used to measure performance, however with some changes:

\[
F = (1 + \beta^2) \frac{\text{accuracy} \cdot \text{decision}}{\beta^2 \cdot \text{accuracy} + \text{decision}}
\]

Keeping beta at 0.5
- Increasing sample size improves performance, but only up to a certain point
- **Saliency** discriminates common n-grams better than entropy
Conclusion & Future Work

- **Microblogging** is an **attractive source** of data for sentiment and opinion mining.
- Authors use **syntactic structures** to describe **emotions** or state facts.
- Some **POS-tags** may be **strong indicators** of **emotional text**.
- **Classifier** is able to **determine** positive, negative and neutral **sentiments** of documents.
- **Plan** to collect a multilingual corpus of Twitter data and compare the characteristics of the **corpus across different languages**.
My Thoughts

• **Part of Speech tagging** is an **effective** way to understand nuances of language and emotion

• Authors **claimed** that their technique is better than previous methods, however, they **never compared** it to previous methods in the entirety of the paper

• Although **conditional independency** can help simplify calculations, it is **too naïve** an assumption

• Another **implicit assumption** is that **people spell words properly** on twitter. Not true.

• Moreover, sarcasm is one of the big problems which cannot be handled by simple sentiment classifiers. **Twitter is full of sarcasm**

• **Test set is too small** (216 data points) compared to train set (300,000 data points!)

• As of 2018, the proposed **future work has not been published** by the main author
References

DEEP CONVOLUTIONAL NEURAL NETWORKS FOR SENTIMENT ANALYSIS OF SHORT TEXTS

John Robert
278822
Seminar Data Analytics I
INTRODUCTION

What is Sentiment analysis?
Sentiment Analysis is also known as opinion mining, it is the process of determining the emotional tone of a series of words

Why short text?
In the era of social media, we express our opinion with limited words

Problem with short text sentiment analysis
Shot text contain limited contextual information, to analyze short text it requires strategies that combine the small text content with prior knowledge and use more than just bag-of-words but also extract information from the sentence/message in a more disciplined way.
What strategy was used?

• Deep convolutional neural network named **Character to Sentence Convolutional Neural Network (CharSCNN)**, with **two convolutional layers** to extract relevant features from words and sentences of any size.

• Two corpora of two different domains: the **Stanford Sentiment Treebank (SSTb)**, which contains sentences from movie reviews; and the **Stanford Twitter Sentiment corpus (STS)**, which contains Twitter messages.
CONVOLUTIONAL NEURAL NETWORK

[Image: Diagram of a convolutional neural network]

INPUT - will hold the raw pixel values the sequence of words in the sentence.

CONVOLUTIONAL LAYER - will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input.

RELU LAYER - will apply an elementwise activation function

POOL LAYER - will perform a downsampling operation along the spatial dimensions (width, height).

FULLY-CONNECTED LAYER - will compute the class scores, where each of the 5 prediction correspond to a class score. CharSCNN computes a score for each sentiment label $\tau \in T$. 
FIRST CONVOLUTIONAL LAYER

• First layer of the network transforms words into real-valued feature vectors (embeddings) that capture morphological, syntactic and semantic information about the words

• We use a fixed-sized word vocabulary $V^{wrd}$

• We use a fixed-sized character vocabulary $V^{chr}$ (words are composed of characters)

• Given a sentences of $N$ words $\{w_1, w_2, \ldots, w_N\}$, every word $w_n$ is converted to a vector $u_n = [r^{wrd}, r^{wch}]$

• Each vector for has a sub-vector, word-level embedding are $r^{wrd} \in \mathbb{R}^{d_{wrd}}$ and the character-level embedding $r^{wch} \in \mathbb{R}^{cl_{0}u}$ for every $w_n$
WORD-LEVEL EMBEDDING

- They are meant to capture **syntactic** (structure of sentences) and **semantic** (relationship between words, phrases, signs and symbols) information.

- We convert a word $w_n$ into its word-level embedding $r^{wrd}$ by using the matrix-vector product $r^{wrd} = W^{wrd} v^w$

- where $v^w$ *is the vector size* is a vector of size $|V^{wrd}|$ which has value 1 at index $w$ and zero in all other positions.

- The matrix $W^{wrd}$ is a parameter to be learned.

---

CHARACTER-LEVEL EMBEDDING

- Used to extract morphological (how words are formed) and shape information of words

- Given a word $w$ composed of $M$ characters $\{c_1, c_2, ..., c_M\}$, we first transform each character $c_m$ into a character embedding $r_m^{\text{chr}}$

- Given a character $c$, its embedding $r^{\text{chr}}$ is obtained by the matrix-vector product $r^{\text{chr}} = W^{\text{chr}} v^c$

- where $v^c$ is a vector of size $|V^{\text{chr}}|$ which has value 1 at index $c$ and zero in all other positions. The input for the convolutional layer is the sequence of character embeddings $\{r_1^{\text{chr}}, r_2^{\text{chr}}, ..., r_M^{\text{chr}}\}$

Convolutional approach to character-level feature extraction.

Course 3111: Seminar: Data Analytics I
Second Convolutional Layer

Why do we use a convolutional layer for sentence-level representation?

Because extracting a sentence-wide feature set have two main problems: sentences have different sizes and; important information in a sentence can appear at any position in the sentence

- Given a sentence $x$ with $N$ words $\{w_1, w_2, ..., w_N\}$, which have been converted to joint word-level and character-level embedding $\{u_1, u_2, ..., u_N\}$
- Extract a sentence-level representation $r_{sent}^x$.
- This layer produces local features around each word in the sentence and then combines them using a max operation to create a fixed-sized feature vector for the sentence.
TRAINING THE NETWORK

• Network is trained by minimizing a negative likelihood over the training set \( D \).
• Given a sentence \( x \), the network with parameter set \( \theta \) computes a score \( s_{\theta}(x)_{\tau} \) for each sentiment label \( \tau \in T \).
• we apply a softmax operation over the scores of all tags \( \tau \in T \):

$$p(\tau|x, \theta) = \frac{e^{s_{\theta}(x)_{\tau}}}{\sum_{\forall \tau \in T} e^{s_{\theta}(x)_{\tau}}}$$

• We use stochastic gradient descent (SGD) to minimize the negative log-likelihood with respect to \( \theta \):

$$\theta \mapsto \sum_{(x,y) \in D} -\log p(y|x, \theta)$$

• where \((x, y)\) corresponds to a sentence in the training corpus \( D \) and \( y \) represents its respective label.
• CharSCNN architecture was implemented using the Theano library
In (Chrupala, 2013), the author proposes a simple recurrent network (SRN) to learn continuous vector representations for sequences of characters, and use them as features in a conditional random field classifier to solve a character level text segmentation and labeling task.

In (Collobert et al., 2011), the authors use a convolutional network for the semantic role labeling task with the goal avoiding excessive task-specific feature engineering. The authors use a similar network architecture for syntactic parsing.
ADVANTAGES OF CharSCNN

• It uses a feed-forward neural network instead of a recursive one and it does not need any input about the syntactic structure of the sentence.

• The addition of one convolutional layer to extract character features.

• CharSCNN approach to extract character-level features is its flexibility.
EXPERIMENT

Dataset

- The movie review dataset used is the recently proposed Stanford Sentiment Treebank (SSTb) which includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences.
- Stanford Twitter Sentiment corpus (STS) the original training set contains 1.6 million tweets that were automatically labeled as positive/negative using emoticons as noisy labels.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Set</th>
<th># sentences / tweets</th>
<th># classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTb</td>
<td>Train</td>
<td>8544</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Dev</td>
<td>1101</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>2210</td>
<td>5</td>
</tr>
<tr>
<td>STS</td>
<td>Train</td>
<td>80K</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Dev</td>
<td>16K</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>498</td>
<td>3</td>
</tr>
</tbody>
</table>
UNSUPERVISED PRE-TRAINING OF WORD-LEVEL EMBEDDINGS

Why unsupervised Learning of Word-Level Embeddings?
• Because recent work has shown that large improvements in terms of model accuracy can be obtained by performing unsupervised pre-training of word embeddings

How was the unsupervised learning of word-level embeddings performed?
• we perform unsupervised learning of word-level embeddings using the word2vec tool3, which implements the continuous bag-of-words and skip-gram architectures for computing vector representations of words
Dataset for pre-training of word embeddings
• December 2013 snapshot of the English Wikipedia corpus as a source of unlabeled data

Processing the dataset
1. removal of paragraphs that are not in English
2. substitution of non-western characters for a special character
3. tokenization of the text using the tokenizer available with the Stanford POS Tagger
4. removal of sentences that are less than 20 characters long (including white spaces) or have less than 5 tokens.
5. we lowercase all words and substitute each numerical digit by a 0 (e.g., 1967 becomes 0000).
Process

• a word must occur at least 10 times in order to be included in the vocabulary, which resulted in a vocabulary of 870,214 entries

• To train our word-level embeddings we use word2vec’s skip-gram method with a context window of size 9.

• The training time for the English corpus is around 1h10min using 12 threads in a Intel Xeon E5-2643 3.30GHz machine
MODEL SETUP

• Development sets is used to tune the neural network hyper-parameters.

• Spent more time was spent tuning the **learning rate** than tuning other parameters, since it is the hyper-parameter that has the largest impact in the prediction performance.

• The only two parameters with different values for the two datasets are the learning rate and the number of units in the convolutional layer that extract sentence features.

• The number of training epochs varies between five and ten for the two dataset.
Neural Network Hyper-Parameters

- we use 4 threads in a Intel Xeon E5-2643 3.30GHz machine.
- Theano based implementation of CharSCNN takes around 10 min. to complete one training epoch for the SSTb corpus with all phrases and five classes.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Name</th>
<th>SSTb</th>
<th>STS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{wrd}$</td>
<td>Word-Level Embeddings dimension</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>$k_{wrd}$</td>
<td>Word Context window</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$d_{chr}$</td>
<td>Char. Embeddings dimension</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$k_{chr}$</td>
<td>Char. Context window</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$c1^0$</td>
<td>Char. Convolution Units</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>$c1_u$</td>
<td>Word Convolution Units</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>$h_{lu}$</td>
<td>Hidden Units</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Learning Rate</td>
<td>0.02</td>
<td>0.01</td>
</tr>
</tbody>
</table>
RESULTS FOR SSTB CORPUS

<table>
<thead>
<tr>
<th>Model</th>
<th>Phrases</th>
<th>Fine-Grained</th>
<th>Positive/Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharSCNN</td>
<td>yes</td>
<td>48.3</td>
<td>85.7</td>
</tr>
<tr>
<td>SCNN</td>
<td>yes</td>
<td>48.3</td>
<td>85.5</td>
</tr>
<tr>
<td>CharSCNN</td>
<td>no</td>
<td>43.5</td>
<td>82.3</td>
</tr>
<tr>
<td>SCNN</td>
<td>no</td>
<td>43.5</td>
<td>82.0</td>
</tr>
<tr>
<td>RNTN (Socher et al., 2013b)</td>
<td>yes</td>
<td>45.7</td>
<td>85.4</td>
</tr>
<tr>
<td>MV-RNN (Socher et al., 2013b)</td>
<td>yes</td>
<td>44.4</td>
<td>82.9</td>
</tr>
<tr>
<td>RNN (Socher et al., 2013b)</td>
<td>yes</td>
<td>43.2</td>
<td>82.4</td>
</tr>
<tr>
<td>NB (Socher et al., 2013b)</td>
<td>yes</td>
<td>41.0</td>
<td>81.8</td>
</tr>
<tr>
<td>SVM (Socher et al., 2013b)</td>
<td>yes</td>
<td>40.7</td>
<td>79.4</td>
</tr>
</tbody>
</table>

- we check whether using examples that are single phrases, in addition to complete sentences, can provide useful information for training
- Phrases indicates whether all phrases (yes) or only complete sentences (no) in the corpus are used for training.
- The Fine-Grained column contains prediction results for the case where 5 sentiment classes (labels) are used (very negative, negative, neutral, positive, very positive).
NOTE

• CharSCNN and SCNN have very similar results in both fine-grained and binary sentiment prediction. These results suggest that the character-level information is not much helpful for sentiment prediction in the SSTb corpus.

• Using phrases as training examples allows the classifier to learn more complex phenomena, since sentiment labeled phrases give the information of how words (phrases) combine to form the sentiment of phrases (sentences).

• Compared to RNTN, CharSCNN has the advantage of not needing the output of a syntactic parser when performing sentiment prediction
RESULTS FOR STS CORPUS

- character-level information has a greater impact for Twitter data
- Using unsupervised pre-training, CharSCNN provides an absolute accuracy improvement of 1.2 over SCNN

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy (unsup. pre-training)</th>
<th>Accuracy (random word embeddings)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharSCNN</td>
<td>86.4</td>
<td>81.9</td>
</tr>
<tr>
<td>SCNN</td>
<td>85.2</td>
<td>82.2</td>
</tr>
<tr>
<td>LProp (Speriosu et al., 2011)</td>
<td></td>
<td>84.7</td>
</tr>
<tr>
<td>MaxEnt (Go et al., 2009)</td>
<td>83.0</td>
<td></td>
</tr>
<tr>
<td>NB (Go et al., 2009)</td>
<td>82.7</td>
<td></td>
</tr>
<tr>
<td>SVM (Go et al., 2009)</td>
<td>82.2</td>
<td></td>
</tr>
</tbody>
</table>
CONCLUSIONS

• positive sentence (left) and its negation (right).
• the extracted features concentrate mainly around the main topic, “film”, and the part of the phrase that indicates sentiment (“liked” and “did ’nt like”).
• left chart that the word “liked” has a big impact in the set of extracted features.
• right chart, we can see that the impact of the word “like” is reduced because of the negation “did ’nt”, which is responsible for a large part of the extracted features.
• negative expression “incredibly dull” is responsible for the features extracted from the sentence in the left
• “definitely not dull”, which is somewhat more positive, is responsible for the features extracted from the sentence in the chart at right.

Why do we use negation?
• negation is an important issue in sentiment analysis
ADVISED IMPROVEMENT

• Use three convolutional layer, first layer for character level embedding, second for word level embedding and third for sentence level embedding

• Perform unsupervised pre-training at character-level representations
REFERENCES

• https://www.brandwatch.com/blog/understanding-sentiment-analysis/
• https://deeplearning4j.org/word2vec.html
• http://cs231n.github.io/convolutional-networks/
• https://www.clickworker.com/2017/03/14/sentiment-analysis-what-is-it-for/
• https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/