Neural Network for Sentiment Analysis

a Tutorial at EMNLP 2016

Yue Zhang and Duy Tin Vo
Singapore University of Technology and Design
Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Outline

- **Introduction**
  - Definition
  - Benchmarks
  - Lexicons
  - Machine learning background

- **Neural Network Background**

- **Sentiment-oriented Word Embedding**

- **Sentence-level Models**

- **Document-level Models**

- **Fine-grained models**

- **Conclusion**
Definition

- Given a set of data $D$: $x^{(i)}$ ($i \leq N$) with label $y^{(i)}$ ($y^{(i)} \leq C$), sentiment analysis task can be deemed as a classification task.
- Extract subjectivity and sentiment polarity from text data.
Definition

- Document Level Sentiment
  - Input
    \[ D = \{d_1, d_2, \ldots, d_n\} \]
    - Where:
      \[ d_i = \{s_1, s_2, \ldots, s_m\} \]
  - Output
    \[ \text{senti} = \{\text{pos, neg, neu}\} \]

This film has everything in it from a jail break, crooked southern politicians, muses, references to what I can only assume are historical figures, riverside baptisms, bank robberies, violence towards animals, singing flocks of religious fanatics, KKK, lynch mobs and so on. There are obviously many references to Homer’s Odyssey in here as well, but I wouldn’t know that because I have never read Homer’s Odyssey or even knew one thing about it. Every other newspaper reviewer seems to know all about it and they think that this cynicism and almost spoof-like quality towards it makes the film that much better. Well coming from a guy who doesn’t know anything about it, I can tell you that it is still an entertaining film. There were times when again, as is usual for a Coen film, I wasn’t sure why I was entertained or laughing, but I was.

This is a road picture where three men travel along the way to find a hidden treasure that Clooney says he has hidden to his two other coi mates. He has to take them along because they were also chained to him when they had their chance to escape.

I like all the principal actors in the film and many of them are Coen cronies. It was nice to see Goodman again. It was nice to see Hunter and especially Turturro who seems to have a place in every Coen film. It’s too bad they didn’t find a place for Steve Buscemi but that is a different story all together. But back to Clooney. The man just has charisma. He is a one hell of an actor as well and here he is not quite as zany as the others but even he has his own idiosyncrasies. His work here is quite awesome and I really hope this shows that he is capable of playing any range of character.

Now after heaping all this praise on the film, let me just say this as well. I didn’t really enjoy the film at first. I found it to be quite tedious and a little boring. There were too many ideas in here and not enough care went into hammering them for all what they were worth. But then the film began to grow on me. It took a while but it did grow on me. I don’t think this is their best film, but it is still a good one and I am giving it a B.5. But the reason that I do recommend this film is for one reason only.

Every day you can go look into the paper and look at the films that are playing and say to yourself, seen it, seen it, oh, seen it last year, that is the same as this film and that is the same as that film. Most films have been recycled in some form or another. Not the Coen’s films. They have not been recycled and if they have I don’t know about it. That is reason enough to see something that they put out. Originality counts for a lot in my books. The Coens are original and they are good. And that is not common in today’s cinema. Enjoy them while they are allowed to make films. Because you don’t get vision like this in many films, so when you do, enjoy it.
Definition

Sentence Level Sentiment

- Input

\[ S = \{s_1, s_2, \ldots, s_n\} \]

- Where:

\[ s_i = \{w_1, w_2, \ldots, w_m\} \]

- Output

\[ senti = \{pos, neg[, neu]\} \]

I like all the principal actors in the film and many of them are Coen cronies.

Definition

- Fine-grained Sentiment
  - Sentiment on target
  - Opinion expression
  - Opinion holder
  - Opinion strength
  - Etc.

Outline

- **Introduction**
  - Definition
  - **Benchmarks**
  - Lexicons
  - Machine learning background

- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Benchmarks

- Movie reviews
    - Subjectivity vs Objectivity sentences
    - Positive vs Negative document

<table>
<thead>
<tr>
<th>Sentence-level</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>subjective</td>
<td>objective</td>
</tr>
<tr>
<td>5000</td>
<td>5000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Document-level</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>negative</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
</tr>
</tbody>
</table>

Subjective:
works both as an engaging drama and an incisive look at the difficulties facing native americans.

Positive:
kolya is one of the richest films i've seen in some time. zdenek sverak plays a confirmed old bachelor ( who's likely to remain so ), who finds his life as a czech cellist increasingly impacted by the five-year old boy that he's taking care of. though it ends rather abruptly-- and i'm whining, 'cause i wanted to spend more time with these characters-- the acting, writing, and production values are as high as, if not higher than, comparable american dramas.
this father-and-son delight-- sverak also wrote the script, while his son, jan, directed-- won a golden globe for best foreign language film and, a couple days after i saw it, walked away an oscar. in czech and russian, with english subtitles.

Benchmarks

- Movie reviews
  - Pang and Lee (2005)
  - Sentence-level

<table>
<thead>
<tr>
<th>Sentence-level</th>
<th>positive</th>
<th>negative</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5331</td>
<td>5331</td>
<td>10662</td>
</tr>
</tbody>
</table>

Positive:

an idealistic love story that brings out the latent 15-year-old romantic in everyone.

Movie reviews

- Mass et al. (2011)
- Document-level

<table>
<thead>
<tr>
<th></th>
<th>pos</th>
<th>neg</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>12500</td>
<td>12500</td>
<td>25000</td>
</tr>
<tr>
<td>Test</td>
<td>12500</td>
<td>12500</td>
<td>25000</td>
</tr>
<tr>
<td>unsup</td>
<td></td>
<td></td>
<td>50000</td>
</tr>
</tbody>
</table>

Positive:
This film has everything in it from a jail break, crooked southern politicians, mules, references to what I can only assume are historical figures, riverside baptisms, bank robberies, violence towards animals, singing flocks of religious fanatics, KKK, lynch mobs and so on. There are obviously many references to Homer’s Odyssey in here as well, but I wouldn’t know that because I have never read Homer’s Odyssey or even knew one thing about it. Every other newspaper reviewer seems to know all about it and they think that this cynicism and almost spoof-like quality towards it makes the film that much better. Well coming from a guy who doesn’t know anything about it, I can tell you that it is still an entertaining film. There were times when again, as is usual for a Coen film, I wasn’t sure why I was entertained or laughing, but I was.

This is a road picture where three men travel along the way to find a hidden treasure that Clooney says he has hidden to his two other cell mates. He has to take them along because they were also chained to him when they had their chance to escape.

I like all the principal actors in the film and many of them are Coen fanatics. It was nice to see Goodman again. It was nice to see Hunter and especially Turturro who seems to have a place in every Coen film. It’s too bad they didn’t find a place for Steve Buscemi but that is a different story all together. But back to Clooney. The man just has charisma. He is a one hell of an actor as well and here he is not quite as zany as the others but even he has his own idiosyncrasies. His work here is quite awesome and I really hope this shows that he is capable of playing any range of character.

Now after heaping all this praise on the film, let me just say this as well. I didn’t really enjoy the film at first. I found it to be quite tedious and a little boring. There were too many ideas in here and not enough care went into laying them out for all of what they were worth. But then the film began to grow on me. It took a while but it did grow on me. I don’t think this is their best film, but it is still a good one and I am giving it a 8.5. But the reason that I do recommend this film is for one reason only.

Every day you can go look into the paper and look at the films that are playing and say to yourself, seen it, seen it, oh, seen it last year, that is the same as this film and that is the same as that film. Most films have been recycled in some form or another. Not the Coen’s films. They have not been recycled and if they have I don’t know about it. That is reason enough to see something that they put out. Originality counts for a lot in my books. The Coens are original and they are good. And that is not common in today’s cinema. Enjoy them while they are allowed to make films. Because you don’t get vision like this in many films, so when you do enjoy it.

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 ACL: HLT, 142-150.
Benchmarks

- Movie reviews
  - Socher et al. (2013), which is induced from Pang and Lee (2005)
  - Phrase-level

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Valid</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>6920</td>
<td>872</td>
<td>1821</td>
</tr>
<tr>
<td>Fine-grained</td>
<td>8544</td>
<td>1101</td>
<td>2210</td>
</tr>
</tbody>
</table>


Product reviews

- Hu and Liu (2004): 5 products
- Fine-grained

[t]

feature[+2]##just received this camera two days ago and already love the features it has.
photo[+2]##takes excellent photos.
night mode[+2]##night mode is clear as day.
use[+1][u]##i have not played with all the features yet, but the camera is easy to use once you get used to it.
viewfinder[-1]##the only drawback is the viewfinder is slightly blocked by the lens.
##however, using the lcd seems to eliminate this minor problem.
camera[+3]##overall it is the best camera on the market.
##i give it 10 stars!
Benchmarks

- Twitter
  - Go et. al. (2009)
  - Sentence-level

<table>
<thead>
<tr>
<th></th>
<th>pos</th>
<th>neg</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>800k</td>
<td>800k</td>
<td>1.6m</td>
</tr>
<tr>
<td>Test</td>
<td>182</td>
<td>177</td>
<td>359</td>
</tr>
</tbody>
</table>

Positive: how can you not love Obama? he makes jokes about himself.
Negative: Naive Bayes using EM for Text Classification. Really Frustrating...
Benchmarks

- Twitter
  - Mitchell et. al. (2013)
  - Open domain

<table>
<thead>
<tr>
<th>Domain</th>
<th>pos</th>
<th>neg</th>
<th>neu</th>
<th>#Sent</th>
<th>#Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>707</td>
<td>275</td>
<td>2,306</td>
<td>2,350</td>
<td>3,288</td>
</tr>
<tr>
<td>Spanish</td>
<td>1,555</td>
<td>1,007</td>
<td>4,096</td>
<td>5,145</td>
<td>6,658</td>
</tr>
</tbody>
</table>

Benchmarks

- Twitter
  - Dong et. al. (2014)
  - Targeted

<table>
<thead>
<tr>
<th></th>
<th>pos</th>
<th>neg</th>
<th>neu</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>1561</td>
<td>1560</td>
<td>3127</td>
<td>6248</td>
</tr>
<tr>
<td>Test</td>
<td>173</td>
<td>173</td>
<td>346</td>
<td>692</td>
</tr>
</tbody>
</table>

**Neutral:**
i hate that i haven't had time for #zbrush in the past two days... we need #zspheres on the [iphone] so i can still sculpt on the go.

Benchmarks

- **Twitter**
  - SemEval13 (Nakov et. al., 2013)
  - Sentence-level

<table>
<thead>
<tr>
<th></th>
<th>pos</th>
<th>neg</th>
<th>neu</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>3662</td>
<td>1466</td>
<td>4600</td>
<td>9729</td>
</tr>
<tr>
<td>Valid</td>
<td>575</td>
<td>340</td>
<td>739</td>
<td>1654</td>
</tr>
<tr>
<td>Test</td>
<td>1573</td>
<td>601</td>
<td>1640</td>
<td>3814</td>
</tr>
</tbody>
</table>

**Positive:** OMG Saturday at 8, p.s. I love you premieres on abc family.

Outline

- **Introduction**
  - Introduction
  - Benchmarks
  - **Lexicons**
    - Machine learning background
- **Neural Network Background**
- **Sentiment-oriented Word Embedding**
- **Sentence-level Models**
- **Document-level Models**
- **Fine-grained models**
- **Conclusion**
Lexicons

- **Manual methods**
  - MPQA lexicon (Wilson et. al., 2005) contains 8222 words

<table>
<thead>
<tr>
<th>Strength</th>
<th>Length</th>
<th>Word</th>
<th>Part-of-speech</th>
<th>Stemmed1</th>
<th>Polarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>type=weaksbj</td>
<td>len=1</td>
<td>word=abandoned</td>
<td>pos1=adj</td>
<td>stemmed1=n</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>type=weaksbj</td>
<td>len=1</td>
<td>word=abandonment</td>
<td>pos1=noun</td>
<td>stemmed1=n</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>type=weaksbj</td>
<td>len=1</td>
<td>word=abandon</td>
<td>pos1=verb</td>
<td>stemmed1=y</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>type=strongsubj</td>
<td>len=1</td>
<td>word=abase</td>
<td>pos1=verb</td>
<td>stemmed1=y</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>type=strongsubj</td>
<td>len=1</td>
<td>word=abasement</td>
<td>pos1=anypos</td>
<td>stemmed1=y</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>type=strongsubj</td>
<td>len=1</td>
<td>word=abash</td>
<td>pos1=verb</td>
<td>stemmed1=y</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>type=weaksbj</td>
<td>len=1</td>
<td>word=abate</td>
<td>pos1=verb</td>
<td>stemmed1=y</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>type=weaksbj</td>
<td>len=1</td>
<td>word=abdicate</td>
<td>pos1=verb</td>
<td>stemmed1=y</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>type=strongsubj</td>
<td>len=1</td>
<td>word=aberration</td>
<td>pos1=adj</td>
<td>stemmed1=n</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>type=strongsubj</td>
<td>len=1</td>
<td>word=aberration</td>
<td>pos1=noun</td>
<td>stemmed1=n</td>
<td>priorpolarity=negative</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8221.</td>
<td></td>
<td>word=zest</td>
<td>pos1=noun</td>
<td>stemmed1=n</td>
<td>priorpolarity=positive</td>
</tr>
</tbody>
</table>

Source: http://sentiment christopherpotts.net/lexicons.html#mpqa

Lexicons

- Manual methods

<table>
<thead>
<tr>
<th>positive</th>
<th>negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>a+</td>
<td>2-faced</td>
</tr>
<tr>
<td>abound</td>
<td>2-faces</td>
</tr>
<tr>
<td>abounds</td>
<td>abnormal</td>
</tr>
<tr>
<td>abundance</td>
<td>abolish</td>
</tr>
<tr>
<td>abundant</td>
<td>abominable</td>
</tr>
<tr>
<td>access</td>
<td>abominably</td>
</tr>
<tr>
<td>able</td>
<td>abominate</td>
</tr>
<tr>
<td>accessible</td>
<td>abomination</td>
</tr>
<tr>
<td>acclaim</td>
<td>abort</td>
</tr>
<tr>
<td>acclaimed</td>
<td>aborted</td>
</tr>
</tbody>
</table>
Lexicons

- Manual methods
  - Mohammad and Turney (2013) Lexicon contains 14182 words with 10 labels (8 emoticons and 2 sentiments)

<table>
<thead>
<tr>
<th></th>
<th>hate</th>
<th>anger</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hate</td>
<td>anticipation</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hate</td>
<td>disgust</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>hate</td>
<td>fear</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>hate</td>
<td>joy</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hate</td>
<td>negative</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>hate</td>
<td>positive</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hate</td>
<td>sadness</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>hate</td>
<td>surprise</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hate</td>
<td>trust</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>anger</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>anticipation</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>disgust</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>fear</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>joy</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>negative</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>positive</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>sadness</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>surprise</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hateful</td>
<td>trust</td>
<td>0</td>
</tr>
</tbody>
</table>

Lexicons

- **Automatic methods**
  - SentiWordNet (Esuli and Fabrizio, 2006) learns positive and negative sentiment scores for synsets in WordNet

<table>
<thead>
<tr>
<th>POS</th>
<th>ID</th>
<th>PosScore</th>
<th>NegScore</th>
<th>SynsetTerms</th>
<th>Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>00001740</td>
<td>0.125</td>
<td>0</td>
<td>able#1</td>
<td>(usually followed by ‘to’) having the necessary means or [...]</td>
</tr>
<tr>
<td>a</td>
<td>00002098</td>
<td>0</td>
<td>0.75</td>
<td>unable#1</td>
<td>(usually followed by ‘to’) not having the necessary means or [...]</td>
</tr>
<tr>
<td>a</td>
<td>00002312</td>
<td>0</td>
<td>0</td>
<td>dorsal#2 abaxial#1</td>
<td>facing away from the axis of an organ or organism; [...]</td>
</tr>
<tr>
<td>a</td>
<td>00002527</td>
<td>0</td>
<td>0</td>
<td>ventral#2 adaxial#1</td>
<td>nearest to or facing toward the axis of an organ or organism; [...]</td>
</tr>
<tr>
<td>a</td>
<td>00002730</td>
<td>0</td>
<td>0</td>
<td>acroscopic#1</td>
<td>facing or on the side toward the apex</td>
</tr>
<tr>
<td>a</td>
<td>00002843</td>
<td>0</td>
<td>0</td>
<td>basiscopic#1</td>
<td>facing or on the side toward the base</td>
</tr>
<tr>
<td>a</td>
<td>00002956</td>
<td>0</td>
<td>0</td>
<td>abducting#1 abducent#1</td>
<td>especially of muscles; [...]</td>
</tr>
<tr>
<td>a</td>
<td>00003131</td>
<td>0</td>
<td>0</td>
<td>adductive#1 adducting#1 adducent#1</td>
<td>especially of muscles; [...]</td>
</tr>
<tr>
<td>a</td>
<td>00003356</td>
<td>0</td>
<td>0</td>
<td>nascent#1</td>
<td>being born or beginning; [...]</td>
</tr>
<tr>
<td>a</td>
<td>00003553</td>
<td>0</td>
<td>0</td>
<td>emerging#2 emergent#2</td>
<td>coming into existence; [...]</td>
</tr>
</tbody>
</table>

Source: [http://sentiment.christopherpotts.net/lexicons.html#sentiwordnet](http://sentiment.christopherpotts.net/lexicons.html#sentiwordnet)

Lexicons

- **Automatic methods**
  - Tang et. al. (2014) consists of 178,781 positive words/phrases and 168,845 negative words/phrases

<table>
<thead>
<tr>
<th>Phrase</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>follow me ... but</td>
<td>-0.592651</td>
</tr>
<tr>
<td>#society</td>
<td>-0.592650</td>
</tr>
<tr>
<td>i can't view</td>
<td>-0.592650</td>
</tr>
<tr>
<td>producer's</td>
<td>-0.592646</td>
</tr>
<tr>
<td>now, i'm</td>
<td>-0.592637</td>
</tr>
<tr>
<td>#although</td>
<td>-0.592631</td>
</tr>
<tr>
<td>twitter like</td>
<td>-0.592629</td>
</tr>
<tr>
<td>a wizard</td>
<td>-0.592627</td>
</tr>
</tbody>
</table>

Outline

- **Introduction**
  - Introduction
  - Benchmarks
  - Lexicons
  - **Machine learning background**
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Machine Learning Background

- **General model:**
  - Train
  - Predict

- **Feature Extractor**
  - Input
  - Features
  - Machine Learning Algorithms
  - Output

- **Manually extract features**

- **Classification Models**

- **One-hot vector**
- **N-grams**
- **Brown Clustering**
- **Lexicons**
- **Patterns**
- **POS**
- **...**
Machine Learning Background

- Neural Network: a sub-area of machine learning
  - Train
  - Predict

Input → Embedding → Features → Classification → Output

Models

Machine Learning Algorithms

Features

Features Embedding Classification
Outline

- Introduction and Background
- **Neural Network Background**
  - Overview
  - Typical Feature Layers
  - Training
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Overview

- General model:
Overview

- **Embedding Layer**
  - Word to vector
  - Look up table

  \[
  \tilde{x}_i = W_{|V|} \times \hat{I}_i
  \]

  - Where:
    - \( \tilde{x}_i \in \mathbb{R}^d \): word embedding
    - \( W_{|V|} \in \mathbb{R}^{|V| \times d} \): embedding matrix
    - \( \hat{I}_i \in \mathbb{R}^{|V|} \): one-hot vector of word \( w_i \)
    - \( d \): embedding dimension

\[s = w_1 \ w_2 \ \ldots \ w_{n-1} \ w_n\]
Overview

Feature Layer
- Automatically learn the representation of inputs
- Matrix-vector multiplication
- Element-wise composition
- Non-linear transformation
Overview

- **Output Layer**
  
  - *Margin output*: \( f_{\text{score}} = W_o \hat{f} + \hat{b}_0 \)
  
  - *Probability output*
    \[
    O_c^{(i)} = P(Y = c | x^{(i)}, \theta )
    = \text{softmax}_c(f_{\text{score}})
    = \frac{e^{\hat{w}^f + b_c}}{\sum_{c'} e^{\hat{w}^f + b_{c'}}}
    \]
  
  - *Predicted label*: \( \hat{y}^{(i)} = \arg\max(O^{(i)}) \)

- *Where*:
  
  - \( \theta \): set of parameters
  
  - \( W_o, \hat{b}_o \): weight and bias parameters of output layer
Outline

- Introduction
- **Neural Network Background**
  - Overview
  - Typical Feature Layers
  - Training
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Typical Feature Layers

- **Feed Forward (MLP)**
  \[ \vec{h}_i = f(W_x \vec{x}_i + \vec{b}_x) \]

  - Where:
    - \( \vec{h}_i \): hidden features
    - \( f(z) \): activation function
    - \( W_x, \vec{b}_x \): weight and bias parameters of MLP
    - \( \vec{x}_i \): input vector

Source: https://www.mql5.com/pt/code/9002
Typical Feature Layers

- **Activation functions** $f(z)$
  - $\text{sigmoid}(z) = \frac{1}{1+e^{-z}}$
  - $\text{tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
  - $\text{softmax}_j(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, j = 1, \ldots, K$
  - $\text{relu}(z) = \max(0, z)$
  - $\text{idem}(z) = z$
**Typical Feature Layers**

- Convolutional neural network (CNN)
  \[ \tilde{c}_i^k = f(\mathbf{W}_c(\tilde{x}_i \oplus \tilde{x}_{i+1} \oplus \ldots \oplus \tilde{x}_{i+k}) + \mathbf{b}_c) \]

  - Where:
    - \( \tilde{c}_i^k \): convolutional features
    - \( f(z) \): activation function
    - \( \mathbf{W}_c, \mathbf{b}_c \): weight and bias parameters of CNN
    - \( \tilde{x}_i \): input vectors
    - \( k \): window size (2,3 in common)
    - \( \oplus \): concatenation

Source: http://parse.ele.tue.nl/education/cluster2
Typical Feature Layers

- **Pooling**
  
  \[ \hat{h}_i = \text{pool}(C_i) \]
  
  - Where:
    - \( \hat{h}_i \): hidden features
    - \( \text{pool} \) is element-wise operations (max, average, min, ...)
    - \( C_i \): input matrix

![Diagram of pooling operations](image)
**Typical Feature Layers**

- Recurrent Neural Network (RNN)
  
  \[ \tilde{h}_i = f(\mathbf{W}_h \tilde{h}_{i-1} + \mathbf{W}_x \tilde{x}_i + \tilde{b}_x) \]

  - Where:
    - \( \tilde{h}_i \): hidden features at time \( i \)
    - \( f(z) \): activation function
    - \( \mathbf{W}_h, \mathbf{W}_x, \tilde{b}_x \): weight and bias parameters of RNN
    - \( \tilde{x}_i \): input vector

  ![Diagram of RNN](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Source: [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
Typical Feature Layers

- Long Short Term Memory (LSTM)
  \[
  \begin{align*}
  \tilde{f}_t &= \sigma(\mathbf{W}_f \tilde{x}_t + \mathbf{U}_f \tilde{h}_{t-1} + \tilde{b}_f) \\
  \tilde{i}_t &= \sigma(\mathbf{W}_i \tilde{x}_t + \mathbf{U}_i \tilde{h}_{t-1} + \tilde{b}_i) \\
  \tilde{u}_t &= \tanh(\mathbf{W}_u \tilde{x}_t + \mathbf{U}_u \tilde{h}_{t-1} + \tilde{b}_u) \\
  \tilde{c}_t &= \tilde{i}_t \odot \tilde{u}_t + \tilde{f}_t \odot \tilde{c}_{t-1} \\
  \tilde{o}_t &= \sigma(\mathbf{W}_o \tilde{x}_t + \mathbf{U}_o \tilde{h}_{t-1} + \tilde{b}_o) \\
  \tilde{h}_t &= \tilde{o}_t \tanh \odot (\tilde{c}_t)
  \end{align*}
\]

- Where:
  - \(\tilde{f}_t, \tilde{i}_t, \tilde{u}_t, \tilde{c}_t, \tilde{o}_t\): forget, input, update, control, output gate layers, respectively
  - \(\mathbf{W}_*, \mathbf{U}_*, \tilde{b}_*\): weight and bias parameters of LSTM

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Typical Feature Layers

- Recursive Neural Network (RecNN)
  \[ \tilde{h}_i = f(W_l \tilde{h}_{i-1}^l + W_r \tilde{h}_{i-1}^r + \tilde{b}_h) \]

  - Where:
    - \( \tilde{h}_i \): hidden features at time \( i \)
    - \( f(z) \): activation function
    - \( W_l, W_r, \tilde{b}_h \): weight and bias parameters of RecNN
Outline

- Introduction
- **Neural Network Background**
  - Overview
  - Typical Feature Layers
  - **Training**
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Training

- Supervised Learning
- Randomly initialized model
- Compare model output with manual reference
Training

- **Loss functions**
  - Cross Entropy Loss (Maximum Likelihood)
    \[
    \mathcal{L}(\theta) = -\frac{1}{N} \sum_i p_i \log(q_i) = -\frac{1}{N} \sum_{i=1}^{N} I_{y^{(i)}} \log(O^{(i)})
    \]
  
  - Where:
    - \(\theta\): set of parameters
    - \(N\): number of samples
    - \(I_{y^{(i)}}\): one-hot vector corresponding to label \(y^{(i)}\)
    - \(O^{(i)}\): probability output of sample \(x^{(i)}\)
Training

- **Loss functions**
  - Hinge loss (maximum-margin)
    - Binary classification:
      \[
      \mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \max(0, 1 - y^{(i)} f_{\text{score}}^{(i)})
      \]
    - Multiclass classification
      \[
      \mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \max(0, 1 + \max_{c \neq c'} f_{\text{score}}_{c'} - f_{\text{score}}_{c})
      \]
  - Where:
    - \(\theta\): set of parameters
    - \(N\): number of samples
    - \(y^{(i)} \in \{-1, 1\}\)
    - \(f_{\text{score}}\): margin output
Training

- Loss functions
  - 0/1 Loss (large margin)
    \[
    \mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} I_{y_i \neq \hat{y}_i}
    \]
  - Where:
    - \( \theta \): set of parameters
    - \( N \): number of samples
    - \( I \): indication function
    - \( y \): ground-true labeled vector
    - \( \hat{y} \): predicted vector
Training

- Loss functions
  - MSE Loss (regression)
    
    \[
    L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2
    \]

  - Where:
    - \( \theta \): set of parameters
    - \( N \): number of samples
    - \( y \) is a ground-true labeled vector
    - \( \hat{y} \) is a predicted vector
Training

- **Back Propagation**
  - **Goal**
    - Find $\frac{\partial L}{\partial \theta}$ for all parameters
    - Adjust parameters accordingly
  - **Derivation**
    - Chain Rule: if $z = f(y)$ and $y = g(x)$, then
      \[
      \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}
      \]
    - Layer-wise calculation
      \[
      \frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}
      \]

Training

- Batch gradient descent is an algorithm in which we repeatedly make small steps downward on an error surface defined by a loss function of a set of parameters over the full training set (N samples)

\[ \theta^{k+1} = \theta^k - \eta \frac{\partial L(\theta)}{\partial \theta} \]

- Where
  - \( \theta \): set of parameters
  - \( \eta \): learning rate

- Problem: N is a very large number
Training

- SGD: Stochastic gradient descent works according to the same principles as batch gradient descent, but proceeds more quickly by estimating the gradient from just one example at a time instead of the entire training set

\[ \theta^{k+1} = \theta^k - \eta \frac{\partial L}{\partial \theta}(\theta, x^{(i)}, y^{(i)}) \]

- Mini-batch SGD (MSGD) works identically to SGD, except that we use more than one training example to make each estimate of the gradient

\[ \theta^{k+1} = \theta^k - \eta \frac{\partial L}{\partial \theta}(\theta, x^{(i:i+n)}, y^{(i:i+n)}) \]

→ Problem: manually adjust learning rate
Training

- Momentum: helps to accelerate SGD in the relevant direction by adding a fraction $\gamma$ of the update vector of the past time step to the current update vector

$$v_k = \gamma v_{k-1} - \eta \frac{\partial L(\theta, x^{(i)}, y^{(i)})}{\partial \theta}$$

$$\theta^{k+1} = \theta^k - v_k$$

Training

- AdaGrad: adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters

\[ \theta^{k+1} = \theta^k - \eta^k g^k \]

- Where:
  - \( g^k \): the gradient of \( \mathcal{L} \) w.r.t \( \theta \) at \( k \)
  - \( \eta^k = \frac{\eta}{\sqrt{\sum_{t=1}^{k} g_t^2 + \epsilon}} \)
  - \( \epsilon \): a smoothing term that avoids division by zero

- Problem: learning rate need to be initialized and gradually shrunk to an infinitesimally small number

Training

- RMSprop*: adjusts the Adagrad method in a very simple way in an attempt to reduce its aggressive, monotonically decreasing learning rate. In particular, it uses a moving average of squared gradients instead

\[ \theta^{k+1} = \theta^k + \Delta \theta^k, \]
\[ \Delta \theta^k = -\frac{\eta}{\text{RMS}[g]_k} g_k \]

- Where:
  - \( \text{RMS} \): root mean square
  - \( \text{RMS}[g]_k = \sqrt{E[g^2]_k + \varepsilon} \), \( E[g^2]_k = \rho E[g^2]_{k-1} + (1 - \rho)g_k^2 \)

*currently unpublished adaptive learning rate method. However, it is usually to cite slide 29 of Lecture 6 of Geoff Hinton’s Coursera class.
Training

- AdaDelta: is an extension of Adagrad to handle the problem of continual decay of learning rates. Instead of accumulating all past squared gradients, it restricts the window of accumulated past gradients to some fixed size $w$

\[
\theta^{k+1} = \theta^k + \Delta \theta^k,
\]

\[
\Delta \theta^k = -\frac{\text{RMS}[\Delta \theta]_{k-1}}{\text{RMS}[g]_k} g_k
\]

- Where:
  - $\text{RMS}$: root mean square
  - $\text{RMS}[\Delta \theta]_{k-1} = \sqrt{E[\Delta \theta^2]_{k-1} + \varepsilon}$, $E[\Delta \theta^2]_{k-1} = \rho E[\Delta \theta^2]_{k-2} + (1 - \rho) \Delta \theta^2_{k-1}$
  - $\text{RMS}[g]_k = \sqrt{E[g^2]_k + \varepsilon}$, $E[g^2]_k = \rho E[g^2]_{k-1} + (1 - \rho) g^2_k$
Training

- Adaptive Moment Estimation (ADAM): is another method that computes adaptive learning rates for each parameter. It is similar to RMSProp with momentum. The simplified ADAM update looks as follows:

\[
\begin{align*}
    m_k &= \beta_1 m_{k-1} - (1 - \beta_1) g_k \\
    v_k &= \beta_2 v_{k-1} - (1 - \beta_2) g_k^2 \\
    \theta^{k+1} &= \theta^k - \eta \frac{m_k}{\sqrt{v_k} + \epsilon}
\end{align*}
\]

Training

- **Regularization**
  - L2: $J(\theta) = \mathcal{L}(\theta) + \lambda \|\theta\|$
    - Where:
      - $\lambda$: decay rate
  - Dropout: $\hat{f}' = \hat{f} \circ r$
    - Where:
      - $r$: a masking vector of Bernoulli random variables with probability $p$ of being 1
  - Batch normalization
  - Rescaling parameters $\theta$ when L2 exceeds a threshold

- **Experimental tricks**
  - OOV: randomly initialization
  - Fine-tune: slightly improve the performance
Outline

- Introduction
- Neural Network Background
  - Sentiment-oriented Word Embedding
    - Overview
    - Traditional word embedding
    - Sentimental-oriented word embedding
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Overview

- Traditional embedding is syntactically and semantically similar, but cannot distinguish sentimental differences.
- How to integrate sentiment information into word embedding
  - Use NN language model to learn syntactic and semantic information
  - Apply labeled data to augment sentiment orientation into word embedding
Outline

- Introduction
- Neural Network Background
- **Sentiment-oriented Word Embedding**
  - Overview
  - Traditional word embedding
    - Sentimental-oriented word embedding
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Traditional Word Embedding

- Unsupervised Learning
  - Basic neural network language models:
    - Input:
      - n-grams
    - Output:
      - probability score of the word given previous words
  - Objective function
    \[
P(w_t | w_1^{t-1}) \approx P(w_t | w_{t-n+1}^{t-1})
    \]
- Problem: probability score
- High computation

Traditional Word Embedding

- **Unsupervised Learning**
  - Pairwise-ranking neural network language models
    - Input: a pair of
      - n-grams: \( t = w_{i-k}w_{i-k+1} \ldots w_i \ldots w_{i+k-1}w_{i+k} \)
      - corrupted n-grams: \( t' = w_{i-k}w_{i-k+1} \ldots w_i' \ldots w_{i+k-1}w_{i+k} \)
    - Output:
      - margin scores \( f(t), f(t') \)
  - Objective function
    \[
    \text{loss}_{cw}(t, t') = \max(0, 1 + f(t') - f(t))
    \]
- Problem: Deep structure
- Still high computation

Traditional Word Embedding

- Unsupervised Learning
  - Simple neural network language models
    - Input:
      - n-grams
    - Output:
      - probability score of the context words given a word or vice versa
  - Objective function
    \[
    \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, c \neq 0} \log P(w_{t+j} | w_t)
    \]
  - Optimization
    - Hierarchical softmax
    - Negative sampling

---

Outline

- Introduction
- Neural Network Background
- **Sentiment-oriented Word Embedding**
  - Overview
  - Traditional word embedding
  - **Sentimental-oriented word embedding**
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Sentimental-Oriented word embedding

• Semi-Supervised Learning
  - Maas et al. (2011) combine an unsupervised probabilistic model and a supervised sentiment component to learn word embedding
  - Objective function
    \[
    v \|R\|_F^2 + \sum_{k=1}^{|D|} \lambda \|\hat{\theta}_k\|_2^2 + \sum_{i=1}^{N_k} \log p(w_i|\hat{\theta}_k; R, b) + \sum_{k=1}^{|D|} \frac{1}{|S_k|} \sum_{i=1}^{N_k} \log p(s_k|w_i; R, \Psi, b_c)
    \]
    • Where:
      - \( p(w_i|\theta; R, b) = \text{softmax}(\theta^T \phi_{w_i} + b) \) \( \rightarrow \) maximum a posteriori (MAP)
      - \( p(s = 1|w_i; R, \psi, b_c) = \sigma(\psi^T \phi_{w_i} + b) \)
      - \( R \in \mathbb{R}^{\beta \times V} : \) word embedding matrix with size of \( \beta \)
      - \( \phi_{w_i} \) is embedding of \( w_i \)
      - \( \theta, \psi, b, b_c : \) weight parameters and bias
      - \( v, \lambda : \) hyper-parameters

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 ACL:HLT, 142-150.
Sentimental-Oriented word embedding

- Supervised Learning
  - Labutov and Lipson (2013) employ pre-trained embedding and labeled data to learn re-embedding words.
  - Objective function

\[
\sum_{d_j \in D} \sum_{w_i \in d_j} \log p(s_j | w_i; \Phi_T) - \lambda \|\Delta \Phi\|_F^2
\]

- Where:
  - \( \Phi_T, \Phi_S : \text{embedding matrices of source and target words} \)
  - \( p(s_j = 1 | w_i; \Phi_T) = \sigma(\psi^T \phi_{w_i} + b) \)
  - \( \Delta \Phi = \Phi_T - \Phi_S \)
  - \( \lambda : \text{hyper-parameter} \)
**Sentimental-Oriented word embeddings**

- SSWE model (Tang et al., 2014)
  - Motivation: $x_{good} \approx x_{bad}$
  - Extend Collobert and Weston (2011) model
  - Adding sentimental information
  - Objective function
    $$loss_{sswe}(t, t^r) = \alpha \times loss_{cw}(t, t^r) + (1 - \alpha) \times loss_s(t, t^r)$$
    
    - Where
      - $loss_{cw}(t, t^r) = \max(0, 1 + f_0(t^r) - f_0(t))$
      - $loss_s(t, t^r) = \max(0, 1 + \delta_s(t)f_1(t^r) - \delta_s(t)f_1(t))$
      - $\delta_s(t) = \begin{cases} 1 & \text{if } f^g(t) = [1,0] \\ -1 & \text{if } f^g(t) = [0,1] \end{cases}$

**Sentimental-Oriented word embeddings**

- **TSWE model** (Ren et al., 2016)
  - **Motivation**
    - Different topics: offensive message vs offensive player
    - Multi-prototype embedding
  - An extension of Tang et al. (2014)
  - Augmenting topical information
  - Objective function
    \[
    loss_{TSWE}(t, t') = \alpha \times loss_{cw}(t, t') + \beta \times loss_t(t, t') + (1 - \alpha - \beta) \times loss_s(t, t')
    \]
    - Where
      - \( loss_{cw}(t, t') = \max(0, 1 + f_0(t') - f_0(t)) \)
      - \( loss_t(t) = -\frac{f(t) \log(softmax(f_{1...N}(t)))}{N} \)
      - \( loss_s(t) = -\frac{f(t) \log(softmax(f_{N+1...M}(t)))}{M} \)

---

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Improving Twitter Sentiment Classification Using Topic-Enriched Multi-Prototype Word Embeddings. In Proceedings of AAAI.
Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- **Sentence-level Models**
  - Overview
  - Bag-of-word methods
  - CNN
  - RecNN
  - RNN
- Document-level Models
- Fine-grained models
- Conclusion
Overview

- Input: a sentence consists of \( n \) words
- Output: polarity or fine-grained sentiment

Classification problem

Classification layer

\[
\hat{y} = \text{softmax}(W_o f + \vec{b}_o)
\]
Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- **Sentence-level Models**
  - Overview
  - **Bag-of-words CNN**
    - RecNN
    - RNN
- Document-level Models
- Fine-grained models
- Conclusion
Bag-of-words

- Bag-of-words (Kalchbrenner et al., 2014)
  - Simply element-wise summing embedding
  - Learning embeddings by back-propagation

Bag-of-words

- Pooling (Tang et. al., 2014; Vo and Zhang, 2015)
  - Make use of Pre-trained word embeddings
  - Extract salient features for traditional classifiers


Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- **Sentence-level Models**
  - Overview
  - Bag-of-word Methods
    - **CNN**
    - RecNN
    - RNN
- Document-level Models
- Fine-grained models
- Conclusion
Convolutional Neural Network

- CNN (Kim, 2014)
  - Feature combinations
  - Single CNN layer
  - Varied-window-size convolutional filters
  - Multichannel (1 static + 1 nonstatic)

Convolutional Neural Network

- Variations
  - dos Santos et al. (2014)
    - Add character information

\[ s = W_1 W_2 \ldots W_n \]
Convolutional Neural Network

Variations

- Kalchbrenner et al. (2014)
  - Fixed-window-size convolutional filters
  - Multiple feature maps
  - K-max, with k dynamically decided
  - Stack multiple convolutional layers

Convolutional Neural Network

- **Variations**
  - Yin and Schütze (2015)
    - Inspired by CNN for RGB kernels in images
    - Employ different kinds of pre-trained embeddings as multichannel
    - Varied-window-size convolutional filters
    - K-max, with k dynamically decided
  - Feature map $F_{i,l}^j$:
    \[
    F_{i,l}^j = \sum_{k=1}^{n} V_{i,l}^{j,k} \ast F_{i-1}^k
    \]
    - Where:
      - $\ast$: the convolution operation
      - $j$: the index of a feature map in layer $i$
      - $V$: a rank 4 tensor weights
Convolutional Neural Network

- Variations
  - Zhang et al. (2016)
    - Make use of different sources of pre-trained embedding with different sizes
    - Employ different sets of convolutional filters

\[
\hat{c}_{ij}^k = f(W_c^j (\hat{x}_i^j \oplus \hat{x}_{i+1}^j \oplus \ldots \oplus \hat{x}_{i+k}^j) + \hat{b}_c^j)
\]
\[
\hat{o}_i^j = pool(C_i^j)
\]

Convolutional Neural Network

- Variations
  - Lei et al. (2015)
    - N-gram tensor
    - Tensor-based feature mapping
    - Non-local
    - Non-linear

\[
z = O^T (P x_1 \odot Q x_2 \odot R x_3)
\]
\[
z[i,j,k] = O^T (P x_i \odot Q x_j \odot R x_k)
\]
Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding

**Sentence-level Models**
- Overview
- Bag-of-word methods
- CNN
- **RecNN**
- RNN

- Document-level Models
- Fine-grained models
- Conclusion
Recursive Neural Network

- RecNN (Socher et al., 2013)

\[ p_1 = f(w[b]) \]
\[ p_2 = f(w[p_1]) \]

Recursive Neural Network

Variations

- Adaptive Multi-Compositionality RecNN (Dong et al., 2014)
  - Employ a set of composition functions

\[
v^i = f \left( \sum_{h=1}^{c} P(g_h|v_i^i, v_r^i)g_h(v_i^i, v_r^i) \right)
\]
\[
\begin{bmatrix}
    P(g_1|v_i^i, v_r^i) \\
    \vdots \\
    P(g_c|v_i^i, v_r^i)
\end{bmatrix} = \beta - \text{softmax} \left( S \begin{bmatrix} v_i^i \\ v_r^i \end{bmatrix} \right)
\]

Recursive Neural Network

- Variations
  - Matrix-Vector RecNN (Socher et al., 2012)
    - Both matrix and vector
    - More composition interaction (Cross-way composition)
    - More features

\[
\begin{align*}
(p_2, P_2) & \quad p_2 = g\left(W\left[\begin{array}{c} C_p^1 \\ P_1 \end{array}\right]\right) \\
(p_1, P_1) & \quad P_2 = W_M\left[\begin{array}{c} P_1 \\ C_1 \end{array}\right]
\end{align*}
\]

Recursive Neural Network

Variations

- Recursive Neural Tensor Network (Socher et al., 2013)
  - Also more composition
  - Less parameters (embeddings)

\[
p_1 = f \left( \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix} \right)
\]

\[
p_2 = f \left( \begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix} \right)
\]

\[p = f \left( \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:2]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix} \right)\]

\[p = f \left( \begin{bmatrix} \end{bmatrix}^T V^{[1:2]} \begin{bmatrix} \end{bmatrix} + W \begin{bmatrix} \end{bmatrix} \right)\]

Problem:
- Extracts non-local features
- Relies on external syntactic parsers for tree structure.

Recursive Neural Network

Variations

- Deep RecNN (Irsoy and Cardie 2014)
  - Stack multiple RecNN layers
    \[ h^{(i)}_{\eta} = f(W^{(i)}_L h^{(i)}_{l(\eta)} + W^{(i)}_R h^{(i)}_{r(\eta)} + V^{(i)} h^{(i-1)}_{\eta} + b^{(i)}) \]
  - Where:
    - \( i \): stacked layer index
    - \( W^{(i)}_L, W^{(i)}_R, V^{(i)}, b^{(i)} \): weight and bias parameters
    - \( l(\eta), r(\eta) \): left and right children of \( \eta \)
Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
  - **Sentence-level Models**
    - Overview
    - Bag-of-word methods
    - CNN
    - RecNN
    - RNN
  - Document-level Models
  - Fine-grained models
- Conclusion
Recurrent Neural Network

- LSTM (Wang et al., 2015)
  - Use a standard LSTM
  - Fine-tune word embeddings

\[
\begin{align*}
\hat{f}_t &= \sigma(W_f \tilde{x}_t + U_f \tilde{h}_{t-1} + \tilde{b}_f), \\
\hat{i}_t &= \sigma(W_i \tilde{x}_t + U_i \tilde{h}_{t-1} + \tilde{b}_i), \\
\tilde{u}_t &= \tanh(W_u \tilde{x}_t + U_u \tilde{h}_{t-1} + \tilde{b}_u), \\
\tilde{c}_t &= \hat{i}_t \odot \tilde{u}_t + \hat{f}_t \odot \tilde{c}_{t-1}, \\
\hat{o}_t &= \sigma(W_o \tilde{x}_t + U_o \tilde{h}_{t-1} + \tilde{b}_o), \\
\tilde{h}_t &= \hat{o}_t \tanh(\tilde{c}_t)
\end{align*}
\]

Source: http://deeplearning.net/tutorial/lstm.html

Recurrent Neural Network

- Variations
  - Bi-directional LSTM: Tai et al. (2015), Li et al. (2015), Teng et al. (2016)

\[ f = h_s^R \oplus h_e^L \]


Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations?. In Proceedings of EMNLP, 2304–2314.

Recurrent Neural Network

- **Variations**
  - Tree Structured LSTM: Tai et al. (2015); Li et al. (2015); Zhu et al. (2015)
    - Child-sum tree \( \rightarrow \) Dependency tree
    - N-ary tree \( \rightarrow \) Constituency tree

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations?. In Proceedings of EMNLP, 2304–2314.
Recurrence Neural Network

- Variations
  - Gated RecNN (Chen et al., 2015)
    - Build a gated structure on the full binary tree

Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- **Document-level Models**
  - Overview
  - Document Embedding
  - Flat Models
  - Hierarchical Learning
- Fine-grained models
Overview

- Input: a document consists of m sentences
- Output: polarity or fine-grained sentiment

Classification problem

Classification layer

\[ \hat{y} = \text{softmax}(W_0 \hat{f} + \hat{b}_0) \]
Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
  - Overview
  - Document Embedding
    - Flat Models
    - Hierarchical Learning
- Fine-grained models
Document Embedding

- Extend Word2vec models (Mikolov et al., 2013) to learn document representations
- Utilize document representation as features for MLP classification


Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- **Document-level Models**
  - Overview
  - Document Embedding
  - **Flat Models**
  - Hierarchical Learning
- Fine-grained models
Flat Models

- **Sentence-level-based models**

- **CNN Variations**
  - Johnson and Zhang (2015a)
    - seq-CNN: use one-hot inputs for a word
    - bow-CNN: use one-hot inputs for n-grams
  - Johnson and Zhang (2015b)
    - Augment inputs by CNN-based region embeddings

- **LSTM Variations**
  - Johnson and Zhang (2016): Extend Jonhson and Zhang (2015b) model by applying LSTM

- One-hot encoding is efficient to represent variable-sized document

---

Flat Models

- Deep CNN Variations
  - Zhang et al. (2015)
    - Use one-hot character-level inputs
    - Stack 6 convolutional layers
  - Conneau et al. (2016)
    - Employ character embeddings
    - Build up to 49 CNN layers

Character-level representation is also helpful

Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- **Document-level Models**
  - Overview
  - Document Embedding
  - Flat Models
    - **Hierarchical Learning**
- Fine-grained models
Hierarchical Learning

- Pooling (Tang et al., 2015a)
  - Average pooling sentence representations as document representation
- LSTM/CNN-GRU (Tang et al., 2015b)


Hierarchical Learning

- Variations
  - LSTM-CNN (Zhang et al., 2016)

Hierarchical Learning

- Variations
  - GRU-GRU Attention networks (Yang et al., 2016)

Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- **Fine-grained models**
  - Overview
  - Targeted Sentiment
  - Open-domain Targeted Sentiment
  - Opinion Expression Detection
- Conclusion
Overview

🔹 Inputs:
  - A sentence consists of n words.
    • With a given target ➔ Classification problem
    • Without a given target ➔ Sequence labeler

🔹 Output:
  - [Who] holds [which opinions] towards [whom]
Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- **Fine-grained models**
  - Overview
  - **Targeted Sentiment**
    - Open-domain Targeted Sentiment
    - Opinion Expression Detection
- Conclusion
Targeted Sentiment

- Tree-structure-based
  - Dong et al. (2014)
    - Variant RecNN
    - Dependency tree

Targeted Sentiment

- Tree-structure-based
  - Nguyen and Shirai (2015)
    - Variant RecNN
    - Dependency+Constituent trees

Targeted Sentiment

- **Pattern-based**
  - Vo and Zhang (2015)
  - Pooling mechanisms

\[ P_{tw} = [F_{tw}^{(1)}, T_{tw}^{(1)}, S_{tw}^{(1)}, F_{tw}^{(2)}, T_{tw}^{(2)}, S_{tw}^{(2)}] \]

Where:
- \( F_{tw}^{(i)} = P(W^{(i)}) \)
- \( T_{tw}^{(i)} = [P(L^{(i)}), P(T^{(i)}), P(R^{(i)})] \)
- \( S_{tw}^{(i)} = [P(LS^{(i)}), P(RS^{(i)})] \)
- \( P(X) = [f_1(X), ..., f_k(X)] \)
- \( f_k \): pooling functions

Targeted Sentiment

- Pattern-based
  - Zhang et al. (2016)
    - Gated mechanisms

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated Neural Networks for Targeted Sentiment Analysis. In Proceedings of AAAI.
Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- **Fine-grained models**
  - Overview
  - Targeted Sentiment
  - **Open-domain Targeted Sentiment**
  - Opinion Expression Detection
- Conclusion
Open-domain Targeted Sentiment

- Open domain (detect target and its sentiment)
  - Zhang et. al. (2015)
    - Neural CRF
    - Discrete features
Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- **Fine-grained models**
  - Overview
  - Targeted Sentiment
  - Open-domain Targeted Sentiment
  - **Opinion Expression Detection**
- Conclusion
Opinion Expression Detection

- Detect opinion expression
  - Irsoy and Cardie (2014)
    - Deep biRNN

The committee, as usual, has refused to make any statements.
Opinion Expression Detection

- Opinion expression and detect target
  - Liu et al. (2015)
    - LSTM
    - Discrete features

Outline

- Introduction
- Neural Network Background
- Sentiment-oriented Word Embedding
- Sentence-level Models
- Document-level Models
- Fine-grained models
- Conclusion
Conclusion

Raw data → Feature Engineering models
Conclusion

Raw data

Neural network models

114
Thank you!!!