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

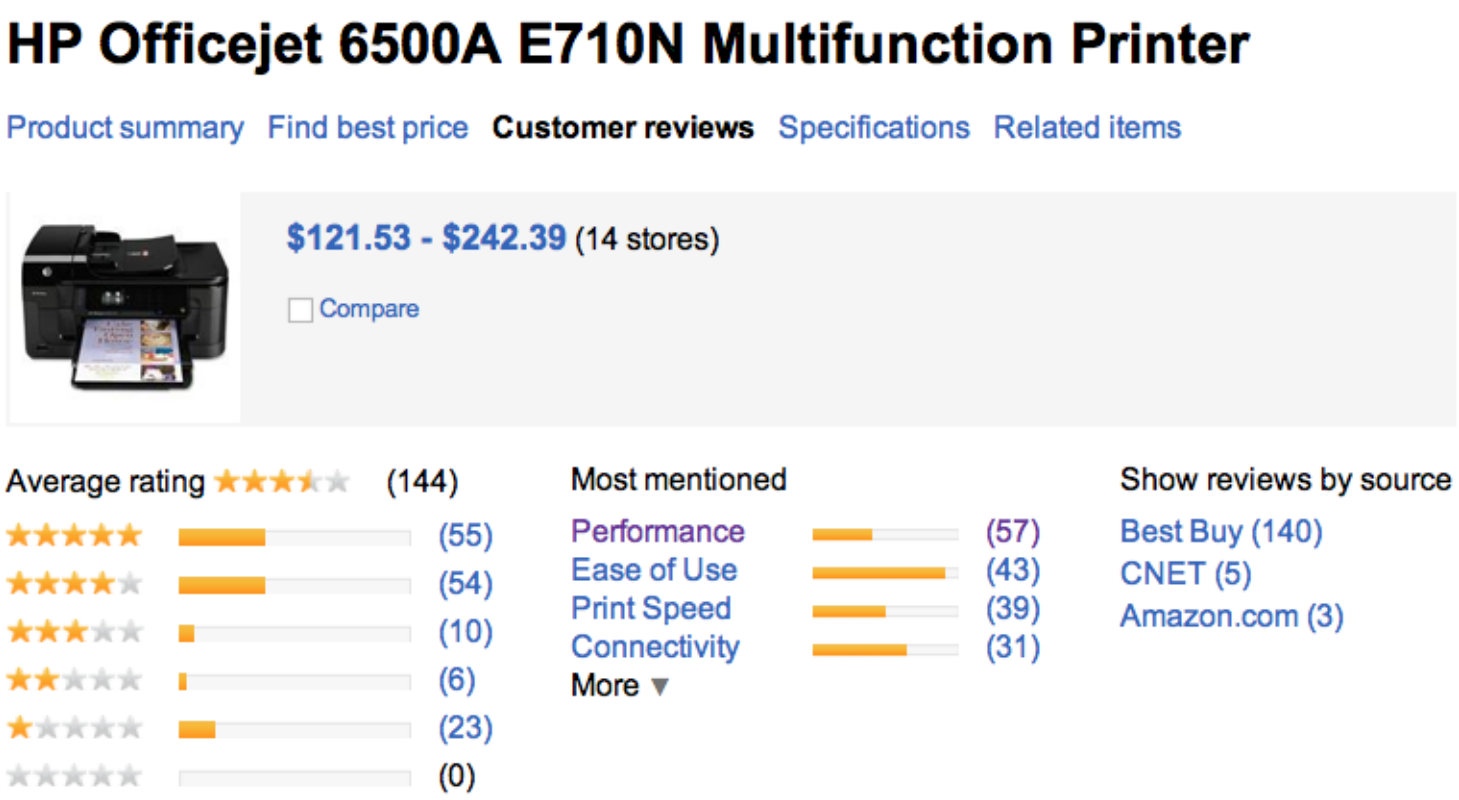
Sentiment Analysis
What is it used **for**?

- 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 policitians' stances and who else supports them
- Social
  - Find like-minded individuals or communities

Bing Shopping



Sentiment Classification in Movie Reviews

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

- 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.

IMDB data in the Pang and Lee database

✓

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)

	Positive List	Negative List	Accuracy	Ties
Human 1	dazzling, brilliant, phenomenal, excellent, fantastic	suck, terrible, awful, unwatchable, hideous	58%	75%
Human 2	gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting	bad, clichéd, sucks, boring, stupid, slow	64%	39%

- Crafted word lists using independent CS grad students
- Positive vs. negative word count

	Positive List	Negative List	Accuracy	Ties
Human 3 + stats	love, wonderful, best, great, superb, still, beautiful	bad, worst, stupid, waste, boring, ?, !	69%	16%

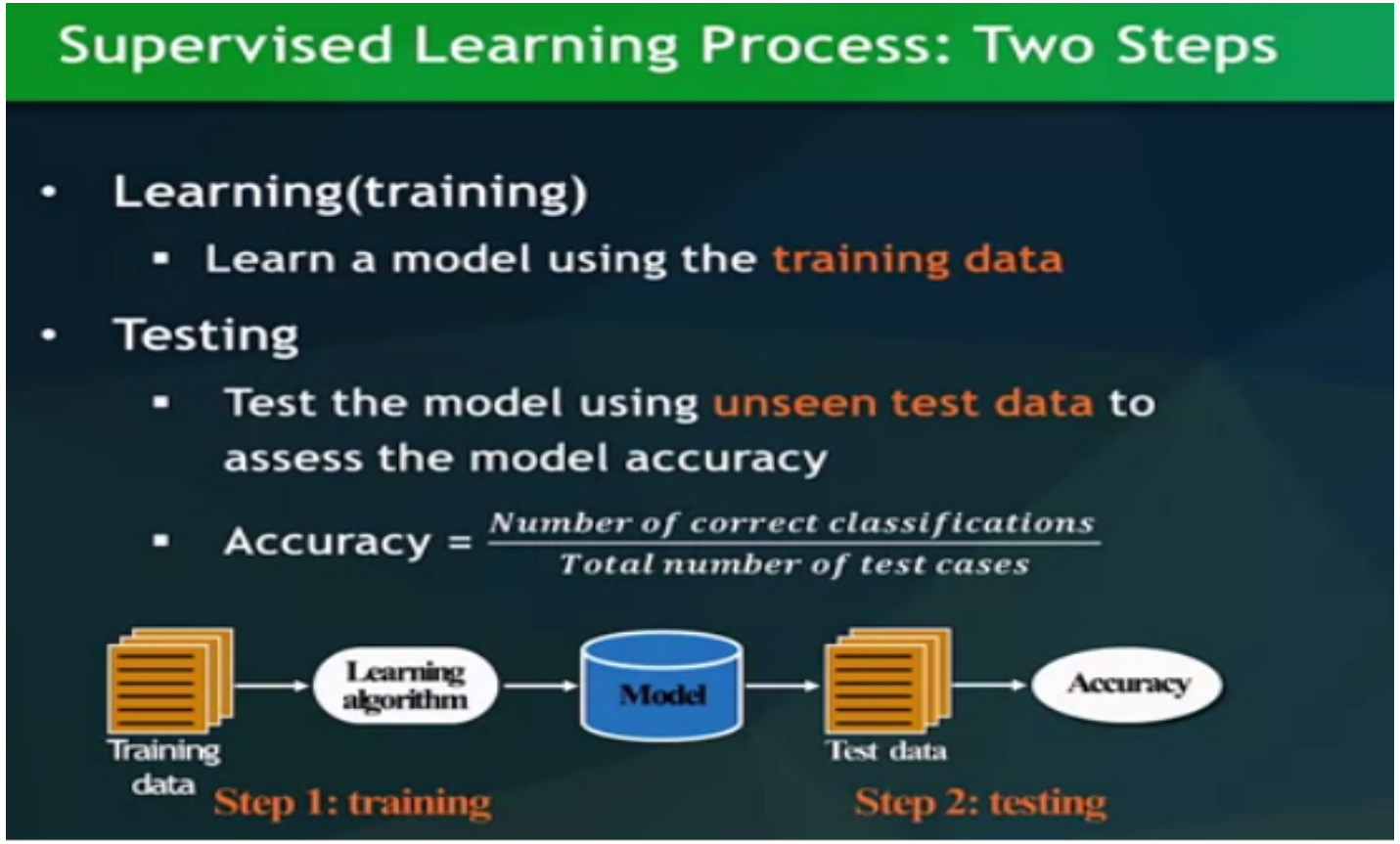
- Frequency counts (including test data)
- Hand-picked words

*Tie rates - Percentage of documents where the two sentiments were rated equally likely.

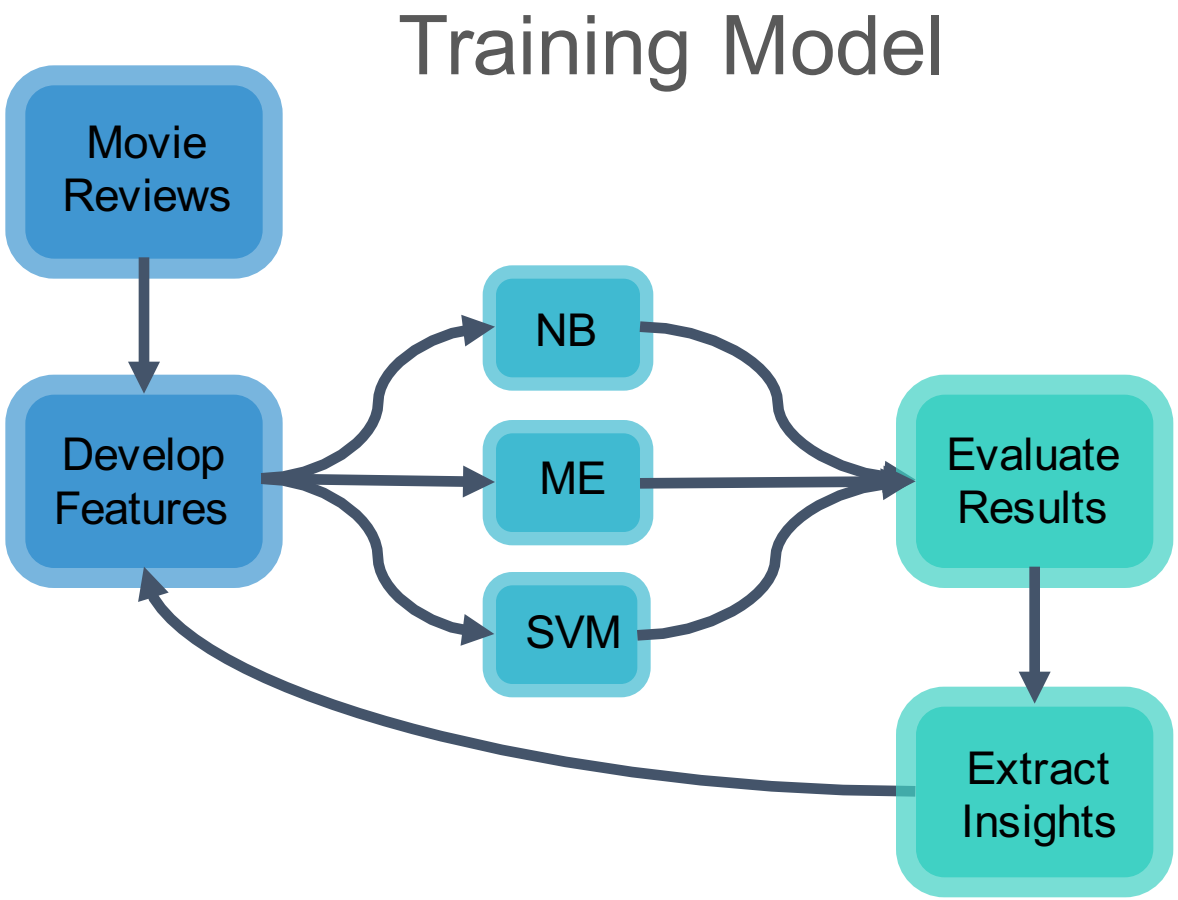
Thumbs up? Sentiment Classification using Machine Learning Techniques

- 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



Framework

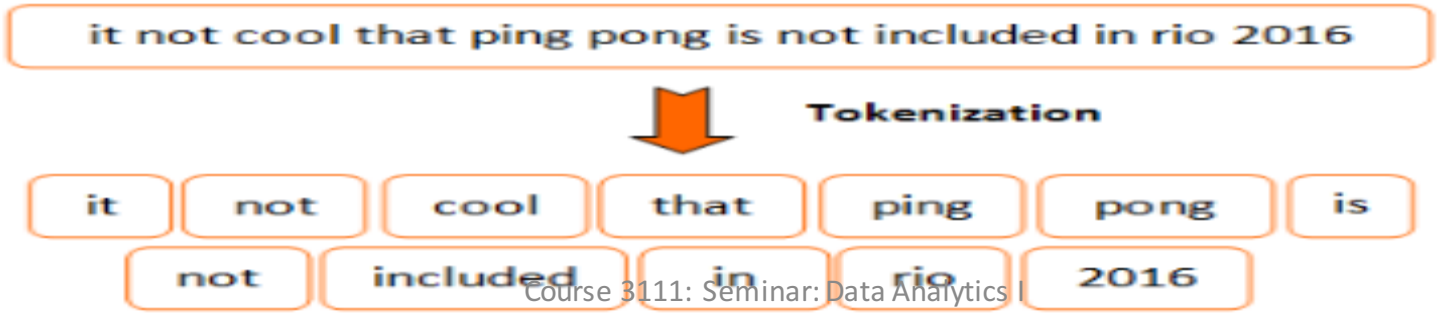

Training Model
Movie Reviews
Develop Features
NB
ME
SVM
Evaluate Results
Extract Insights

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

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

Naïve Bayes Classifier will be - :

$$c^* = \arg max c \, \mathrm{P}(c \mid d)$$

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 maxcP(c|d).

Assume all features are conditionally independent

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

ni(d) is the number of times fi occurs in document d.

$$P(Sentiment \mid Sentence) = \frac{P(Sentence \mid Sentiment)P(Sentiment)}{P(Sentence)}$$

$$P(Word \mid Sentiment) = \frac{Numberofwordsoccurenceinclass + 1}{No.ofwordsbelongingtoaclass + Tota\ln oofwords}$$

Example

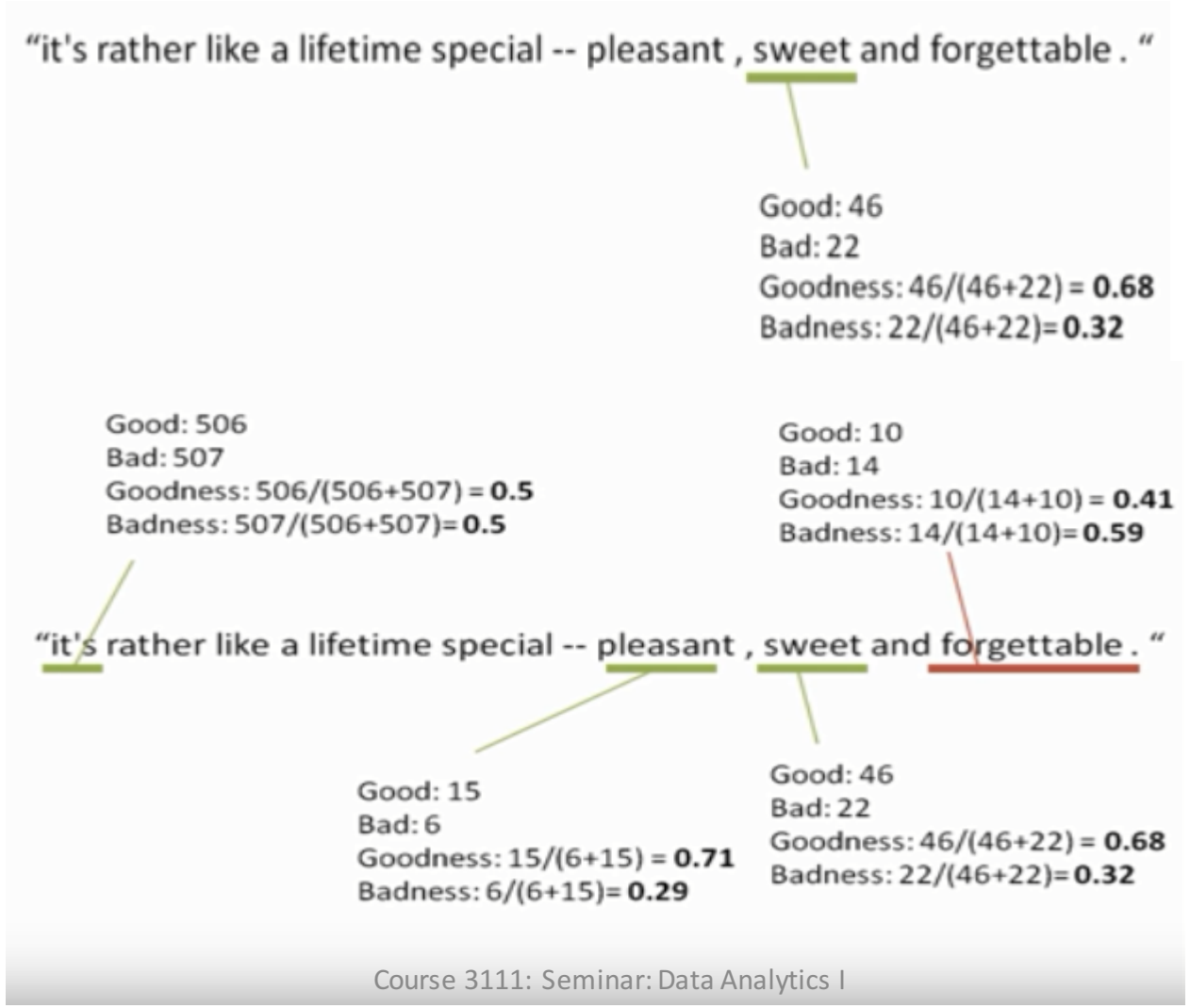
POSITIVE POLARITY (GOOD)

-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 .

-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 .

NEGATIVE POLARITY (BAD)

Example(Continued…)

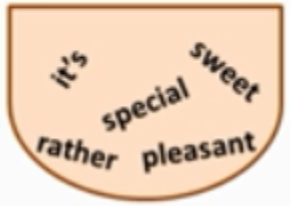


Example(Continued…)

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

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

"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:

$$P_{\mathrm{ME}}(c \mid d) := \frac{1}{Z(d)} \exp\left(\sum_i \lambda_{i,c} F_{i,c}(d,c)\right) ,$$

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

Z(d) is a normalization function. The $\lambda_{i,c}$'s are feature-weight parameters. Larger $\lambda_{i,c}$'s means fi is considered a strong indicator for class c.

Support Vector Machine

Find a hyperplane that makes the margin between two categories.

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

$$z_j = w_1 x_{j1} + w_2 x_{j2} + \ldots\ldots + w_d x_{jd} + b$$

$$\text{i.e. } z_j = x_j^T w + b$$

where,

xi is a p-dimensional real vector.

w is vector that contains the weights and is given as

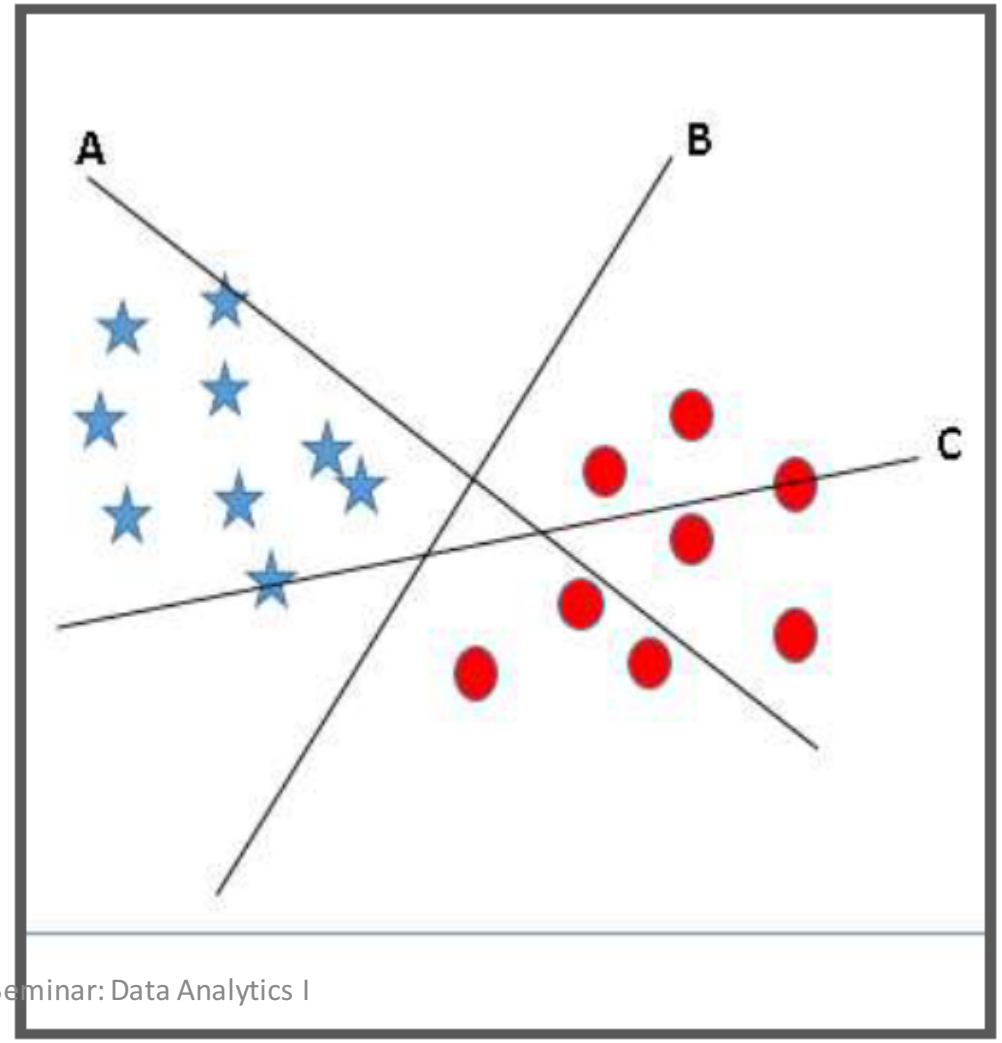
$$\vec{w} = \sum_j \alpha_j c_j \vec{d}_j, \quad \alpha_j \geq 0, \; c_j = \{1,-1\}$$

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.



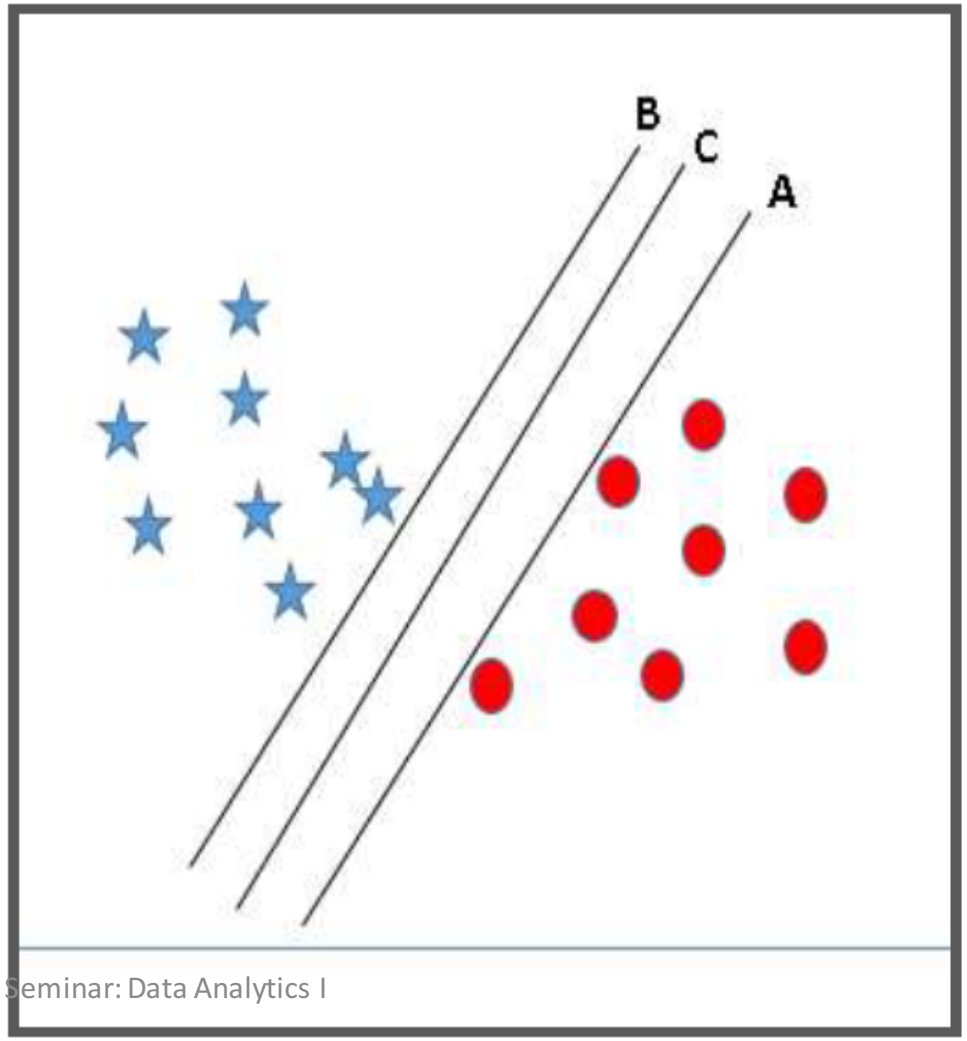
Scenario 2

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 hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**

Above, you can see that the margin for hyper-plane 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 hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.



Results

- Results for different feature:

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

- 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

- B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment classification using machine learning techniques,” in Proc Conf on Empirical Methods in Natural Language Processing (EMNLP), pp. 79–86, 2002.
- Esuli A, Sebastiani F. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In: Proc of LREC 2006 - 5th Conf on Language Resources and Evaluation, 2006.
- Zhang E, Zhang Y. UCSC on TREC 2006 Blog Opinion Mining. TREC 2006 Blog Track, Opinion Retrieval Task.
- Devitt A, Ahmad K. Sentiment Polarity Identification in Financial News: A Cohesion-based Approach. ACL 2007.
- Bo Pang , Lillian Lee, A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts, Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, p.271-es, July 21-26, 2004.


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:

$$P^T_{1,2} = \frac{N_1^T - N_2^T}{N_1^T + N_2^T}$$

- Where N_i is the number of occurences of tag T in set *I*
- Compared two sets of data:
  i. Subjective vs. Objective
  ii. Negative vs. Positive

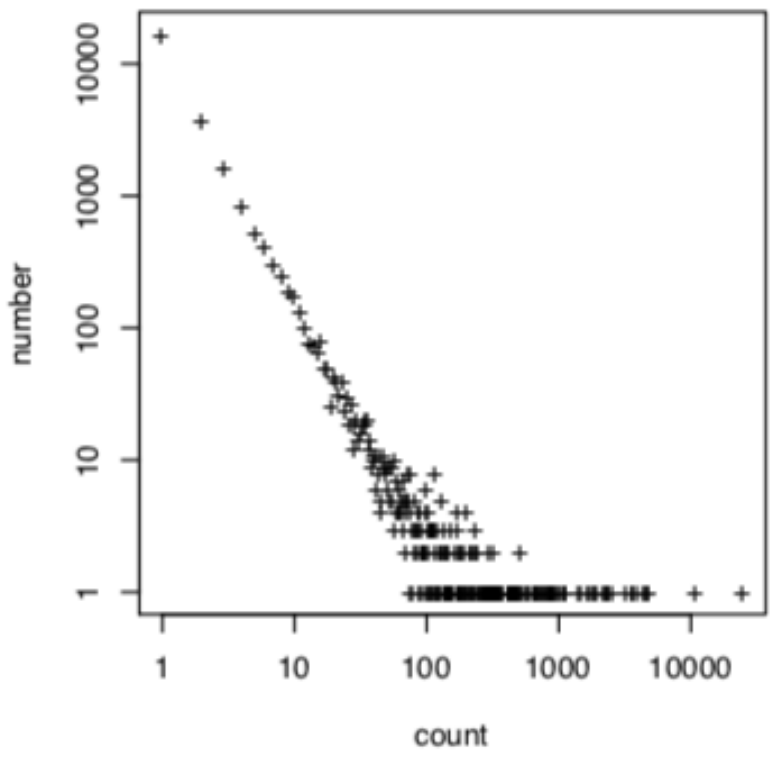


Figure 1: The distribution of the word frequencies follows Zipf's law

TreeTagger

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

word	pos	lemma
The	DT	the
TreeTagger	NP	TreeTagger
is	VBZ	be
easy	JJ	easy
to	TO	to
use	VB	use
.	SENT	.

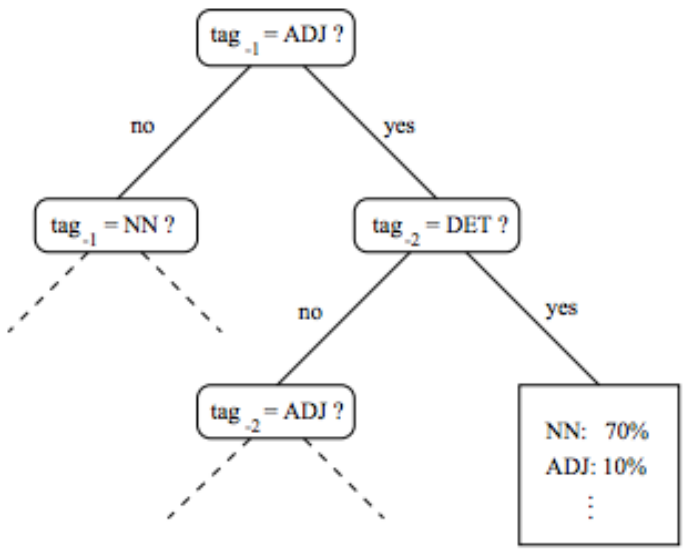


Figure 1: A sample decision tree

Corpus Analysis

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

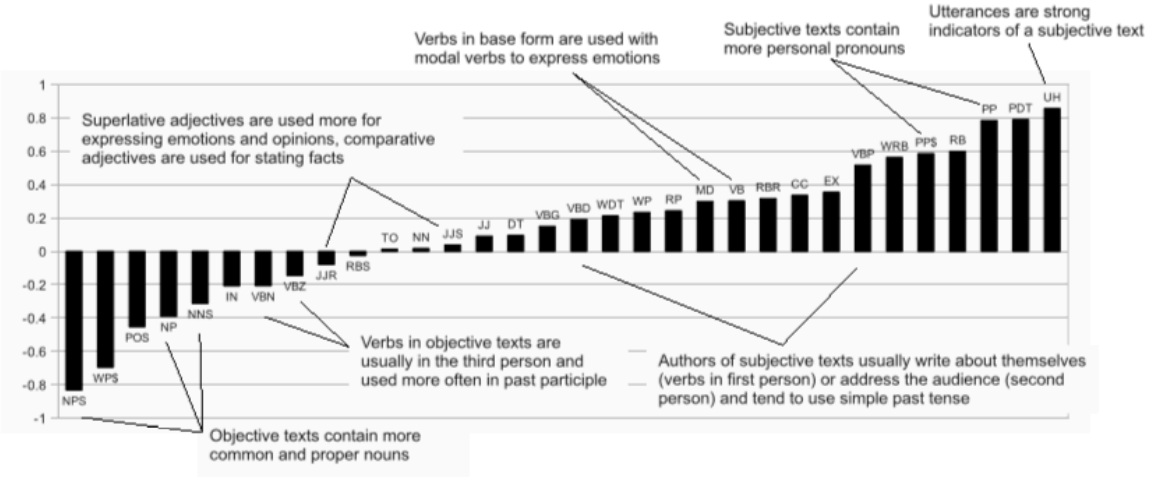


Figure 2: P^T values for objective vs. subjective

Positive	Negative
Superlative adverbs and possesive endings (POS)	Verbs in past tense (VBN, VBD)

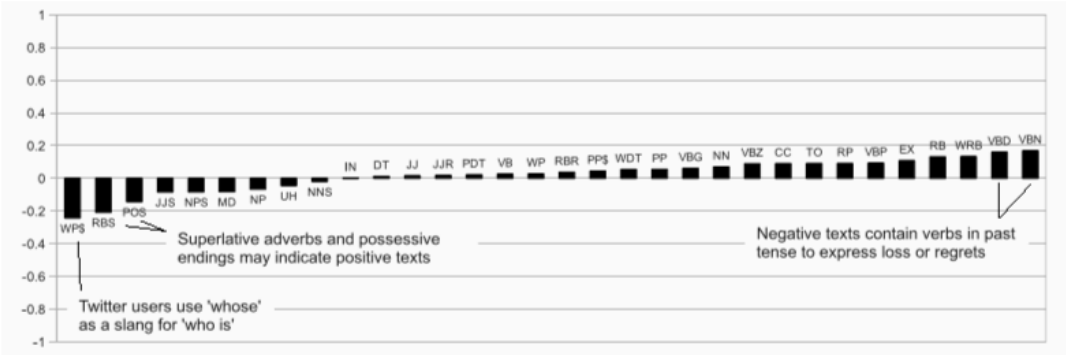


Figure 3: P^T values for positive vs. negative

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 Preperation:**
  - **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

$$P(s|M) = \frac{P(s) \cdot P(M|s)}{P(M)} \longrightarrow \quad P(s|M) = \frac{P(M|s)}{P(M)} \longrightarrow \quad P(s|M) \sim P(M|s)$$

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 θ or salience lower than θ

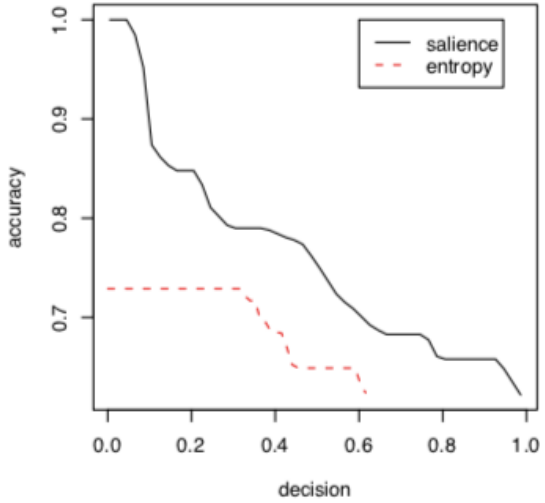
Model Validation

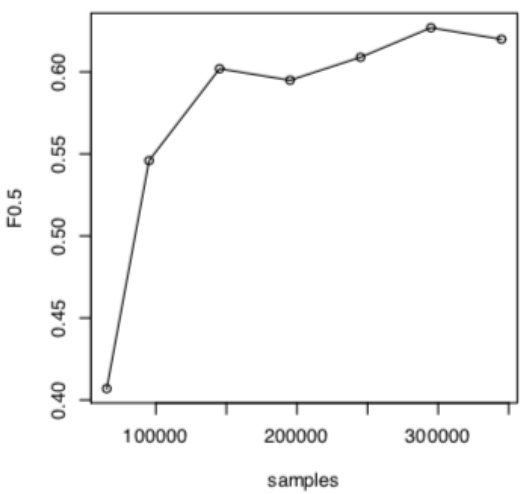
- Classifier was tested on a set of real hand-annotated twitter posts

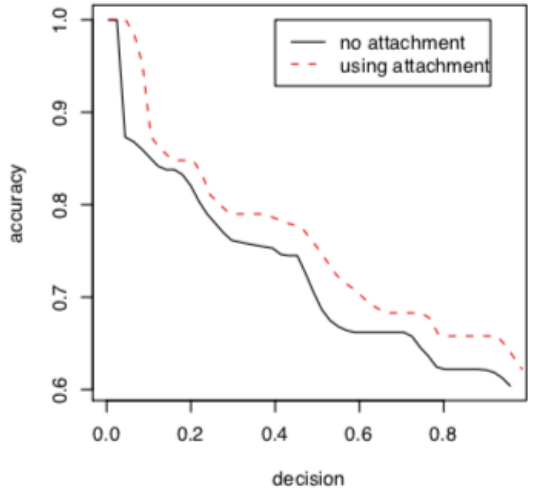
$$accuracy = \frac{N(\text{correct classifications})}{N(\text{all classifications})}$$

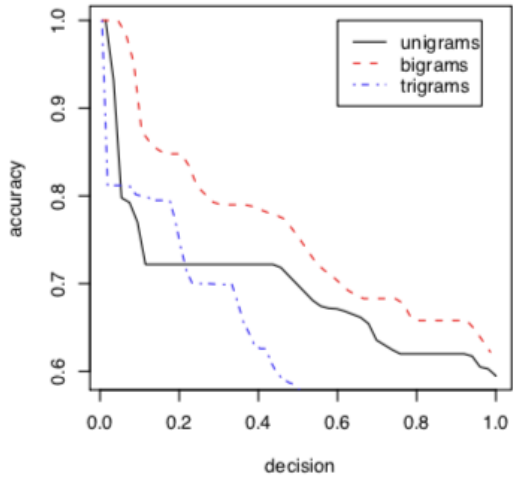
$$decision = \frac{N(\text{retrieved documents})}{N(\text{all documents})}$$

Sentiment	Sample size
Positive	108
Negative	75
Neutral	33
Total	216









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:

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

Keeping beta at 0.5

- Increasing sample size improves performance, but only up to a certain point
- **Salience** 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

- G. Adda, J. Mariani, J. Lecomte, P. Paroubek, and M. Raj- man. 1998. The GRACE French part-of-speech tagging evaluation task. In A. Rubio, N. Gallardo, R. Castro, and A. Tejada, editors, *LREC*, volume I, pages 433–441, Granada, May.
- Ethem Alpaydin. 2004. *Introduction to Machine Learn- ing (Adaptive Computation and Machine Learning)*. The MIT Press.
- Hayter Anthony J. 2007. *Probability and Statistics for En- gineers and Scientists*. Duxbury, Belmont, CA, USA. Kushal Dave, Steve Lawrence, and David M. Pennock.
- 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 519–528, New York, NY, USA. ACM.
- Alec Go, Lei Huang, and Richa Bhayani. 2009. Twit- ter sentiment analysis. Final Projects from CS224N for Spring 2008/2009 at The Stanford Natural Language Processing Group.
- Bernard J. Jansen, Mimi Zhang, Kate Sobel, and Abdur Chowdury. 2009. Micro-blogging as online word of mouth branding. In *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3859–3864, New York, NY, USA. ACM.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Christopher D. Manning and Hinrich Schu¨ tze. 1999. *Foun- dations of statistical natural language processing*. MIT Press, Cambridge, MA, USA.
- Bo Pang and Lillian Lee. 2008. Opinion mining and senti- ment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135. Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan.
- 2002. Thumbs up? sentiment classification using ma- chine learning techniques. In *Proceedings of the Con- ference on Empirical Methods in Natural Language Pro- cessing (EMNLP)*, pages 79–86.
- Ted Pedersen. 2000. A simple approach to building en- sembles of naive bayesian classifiers for word sense dis- ambiguation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguis- tics conference*, pages 63–69, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Jonathon Read. 2005. Using emoticons to reduce depen- dency in machine learning techniques for sentiment clas- sification. In *ACL*. The Association for Computer Lin- guistics.
- Helmut Schmid. 1994. Probabilistic part-of-speech tag- ging using decision trees. In *Proceedings of the Inter- national Conference on New Methods in Language Pro- cessing*, pages 44–49.
- Claude E. Shannon and Warren Weaver. 1963. *A Mathe- matical Theory of Communication*. University of Illinois Press, Champaign, IL, USA.
- Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level senti- ment analysis. In *HLT '05: Proceedings of the con- ference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347– 354, Morristown, NJ, USA. Association for Computa- tional Linguistics.
- Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Emotion classification using web blog corpora. In *WI '07: Proceedings of the IEEE/WIC/ACM Interna- tional Conference on Web Intelligence*, pages 275–278, Washington, DC, USA. IEEE Computer Society.

DEEP CONVOLUTIONAL NEURAL NETWORKS FOR SENTIMENT ANALYSIS OF SHORT TEXTS

John Robert

278822

Seminar Data Analytics I

OUTLINE

1. Introduction
2. Convolutional neural network architecture
3. Training the network
4. Related works
5. Advantages of CharSCNN
6. Experiment
7. Model setup
8. Results
9. Conclusions
10. Advised improvement
11. References

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

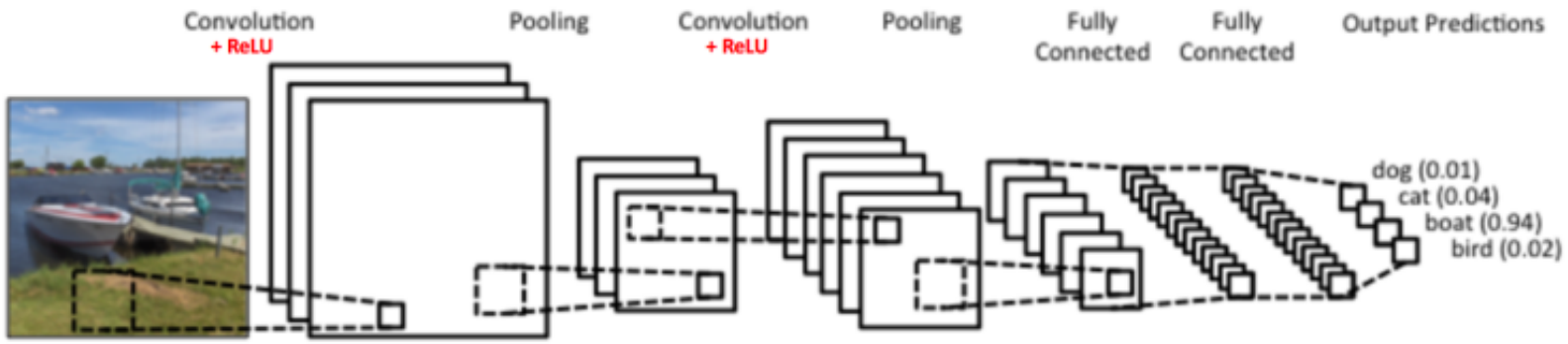
Shot text contain limited contextual information, to analyze short text it requires strategies that combine the small text content with prior knowledge and use more than just bag-of-words but also extract information from the sentence/message in a more disciplined way.

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



https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

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 $\tau \in T$.

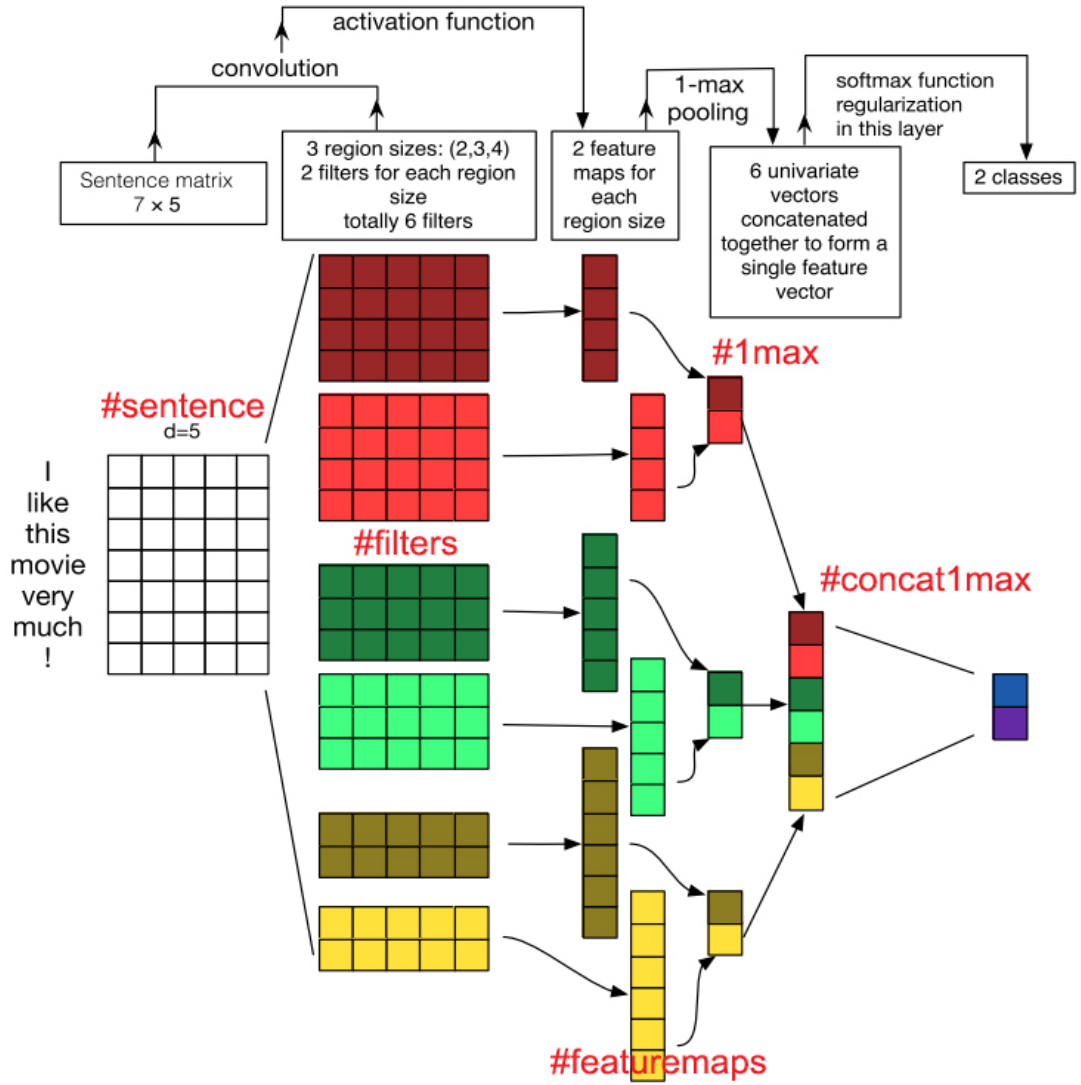
FIRST CONVOLUTIONAL LAYER

- 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

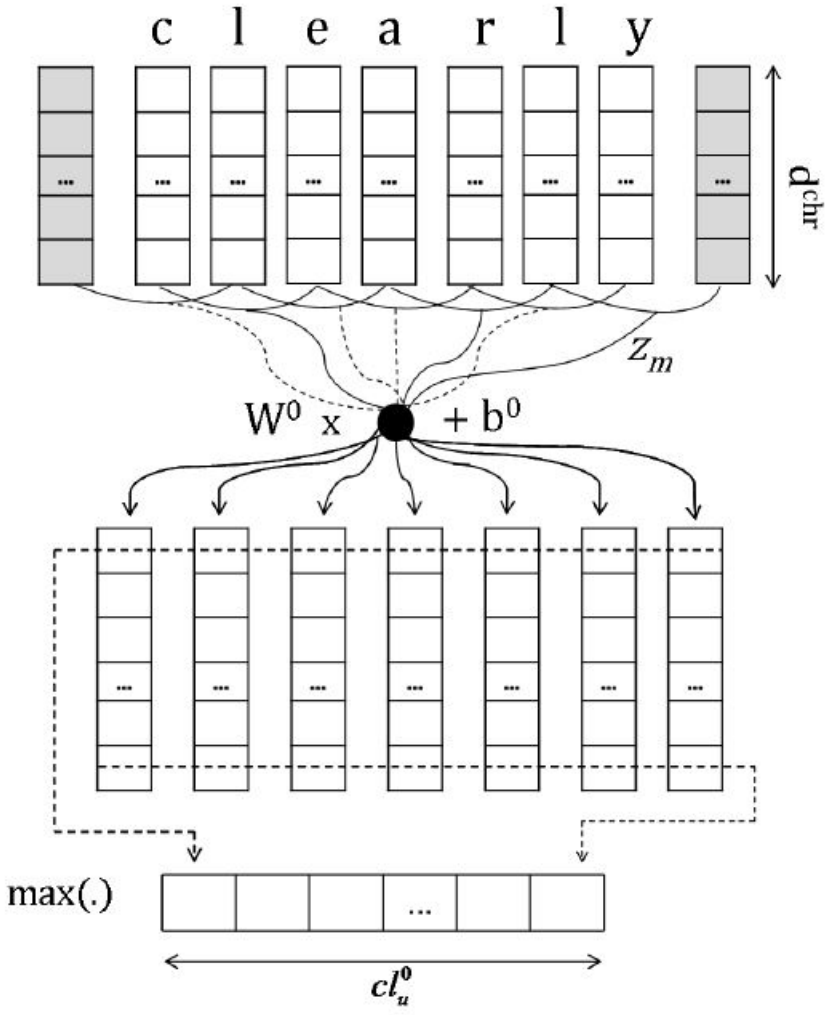


http://www.joshuakim.io/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/

- 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



Convolutional approach to character-level feature extraction.

- Used to extract **morphological**(how words are formed) and **shape** information of words
- Given a word w composed of M characters $\{c_1, c_2, \ldots, c_M\}$, we first transform each character c_m into a character embedding r_m^{chr}
- Given a character c, its embedding r^{chr} is obtained by the matrix-vector product $r^{chr} = W^{chr} v^c$
- where v^c is a vector of size $|V^{chr}|$ 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 $\{r_1^{chr}, r_2^{chr}, \ldots, r_M^{chr}\}$

SENTENCE-LEVEL REPRESENTATION

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 $\{w_1, w_2, \ldots, w_N\}$, which have been converted to joint word-level and character-level embedding $\{u_1, u_2, \ldots, u_N\}$
- 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 θ 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 $\tau \in T$. $p(\tau|x,\theta) = \frac{e^{s_\theta(x)_\tau}}{\sum_{\forall i \in T} e^{s_\theta(x)_i}}$
- We use stochastic gradient descent (SGD) to minimize the negative log-likelihood with respect to θ :

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

- 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

RELATED WORKS

- 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 it 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

Dataset	Set	# sentences / tweets	# classes
SSTb	Train	8544	5
	Dev	1101	5
	Test	2210	5
STS	Train	80K	2
	Dev	16K	2
	Test	498	3

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 was 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.

Parameter	Parameter Name	SSTb	STS
d^{wrd}	Word-Level Embeddings dimension	30	30
k^{wrd}	Word Context window	5	5
d^{chr}	Char. Embeddings dimension	5	5
k^{chr}	Char. Context window	3	3
cl_u^0	Char. Convolution Units	10	50
cl_u^1	Word Convolution Units	300	300
hl_u	Hidden Units	300	300
λ	Learning Rate	0.02	0.01

RESULTS FOR SSTB CORPUS

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

- 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

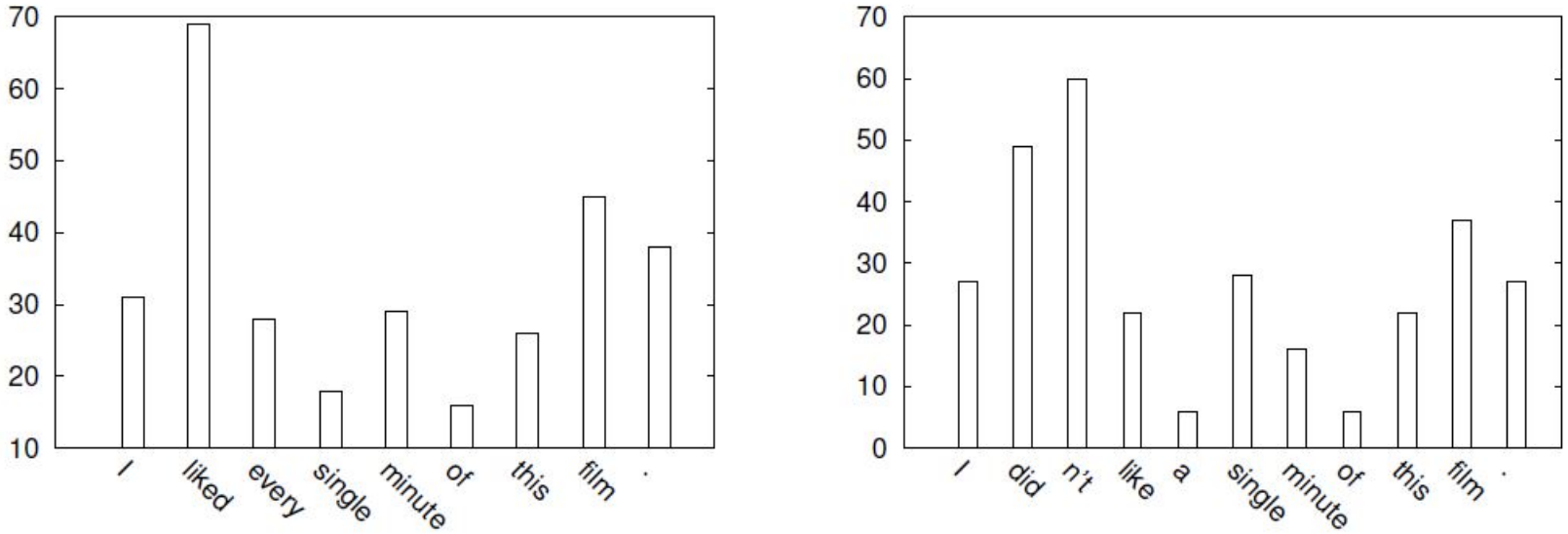
- CharSCNN and SCNN have very similar results in both fine-grained and binary sentiment prediction. These results suggest that the character-level information is not much helpful for sentiment prediction in the SSTb corpus.
- Using phrases as training examples allows the classifier to learn more complex 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

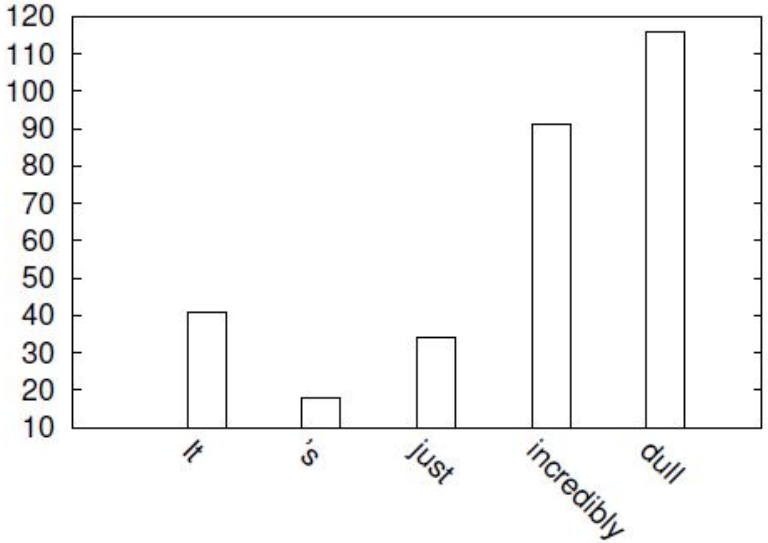
- character-level information has a greater impact for Twitter data
- Using unsupervised pre-training,CharSCNN provides an absolute accuracy improvement of 1.2 over SCNN

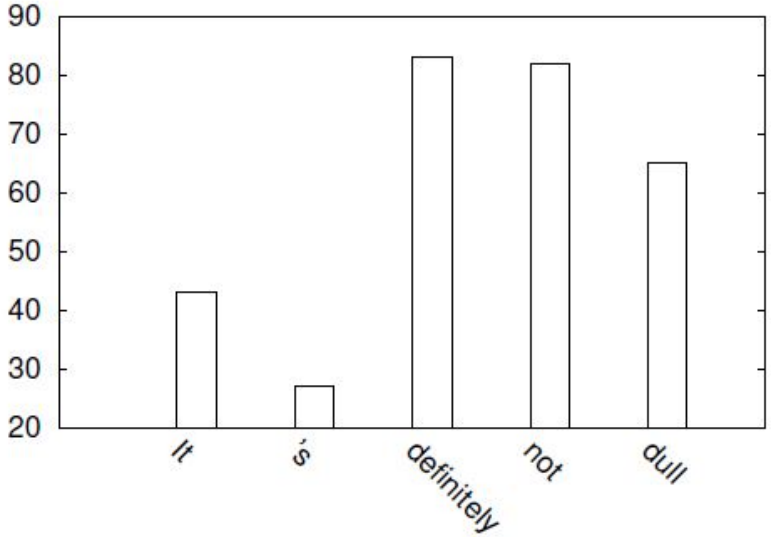
Model	Accuracy (unsup. pre-training)	Accuracy (random word embeddings)
CharSCNN	**86.4**	81.9
SCNN	85.2	82.2
LProp (Speriosu et al., 2011)	84.7	
MaxEnt (Go et al., 2009)	83.0	
NB (Go et al., 2009)	82.7	
SVM (Go et al., 2009)	82.2	

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/
- https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/