Topic Specific Sentiment Analysis with Lexical and Grammatical Modeling

Tianshuo Deng Maochen Guan Yaxing Chen
Computer Science Department
New York University
{td859,mg3364,yc1116}@nyu.edu

Abstract

Sentiment analysis for twitter requires a different approach comparing to news text on the web. Most researches use single model and treat tweet as bag of words. Using parsing technique on tweet is hard due to its nature of informality and noisiness. In this paper, we discuss how to apply sentiment analysis and parsing on tweets. we built a two level model that combines lexical and grammatical model. In lexical model, we treat tweets as bag of words and achieves relatively high recall but low precision. In grammatical model, on the other side, we parse the tweet and build a model with relatively high precision but low recall. Finally, we combines two models together to show how the combined model outperforms both lexical model and grammatical model.

1 Introduction

Tweets are of heavy interests among researchers in text mining area for its unique natures. Within each tweet, users idea is often concisely presented due to its 140 characters limitation. Comparing to news text, it provides in-time information, people post tweet from their mobile device frequently. As a result, most events, news appear on twitter at first time very often. Thus, analysing tweets provides very valuable information.

In this project, we built a system (www.tweetemotion.com) that monitors tweets for certain topic and classify them to POSITIVE, NEGATIVE and NEUTRAL based on their sentiment. For example, given "I love apple" and the topic word "apple", the tweet should be classified as "positive". Intuitively, sentiment is attached to a specific topic, so "I like apple but I hate microsoft" should be classified as "positive" for "apple" and as "negative" for "microsoft".

Features can be captured by analyzing words in a tweet. Traditional n-gram method can be used, [Pang and Lee(2008)]. But it fails to capture the features related to the topic. To fix this, in this paper [Jiang et al.(2011)Jiang, Yu, Zhou, Liu, and Zhao] from Microsoft, introduced a way of finding words nearly topic word as features by some predefined rules. Building such rules requires comprehensive knowledge. Also, from our observations, the words that appear near the topic do not always provide concrete and correct information, as discussed in Section[3.2.1]

To capture the sentiment information precisely, our work is based on two ideas. First, tweet are treated as bag of words which performs well when tweet are irregular, for example("Oooo #iphone–#epicFail"). However, it does not work well for those noisy features in the training data or when more sentiment meaning appeared in same sentence. Therefore in Section[3.2], we built a grammatical model which discovers relationships and dependencies among words. Grammatical model is able to capture sentiment features towards the specific topic. Due to the informality of tweet, parsing is a challenging task, in section[2.4] and Section[3.2.7] we discuss how to preprocess and prune tweets.

Both models have its own advantage, in Section[3.3] and Section[3.4] we discuss how to combine two models together using backoff rules as well as interpolation. The combined model surpass the performance of either lexical model or grammatical model.

2 Experimental Setup

2.1 Data Collection

There are two data sets used in the project. One is the Senders Analytics whereas another one is fetched from Twitter.com using simple rules.

1. Senders Analytics (http://www.sananalytics.com) Senders Analytics contains 2891 tweets. All of the tweets are manually classified by human.Those tweets are directly pulled from Twitter without filtering. The following is an example excerpt from Senders Analytics: Damn it, listening to apple Siri is making me want to upgrade my current cellular! [Tagged as Positive]

   The following topics are covered in the Senders Analytics: Apple, Google, Microsoft, Twitter.

2. Data filtered by pre-defined rules This data set contains 1862 tweets which are retrieved from twitter.com by selected topics and pre-defined rules listed below.

   (a) Create a list that contains both positive and negative words manually.

   (b) If a tweet contains only positive words and there is no stop word between positive and the topic then classify it to positive training samples. If

a tweet contains only negative words and there is no stop word between positive and the topic then we classify it to negative training samples.

From our observation, those rules filtered the tweets with high precision. The following is an example:

Listing 1: positive tweet

```
I really like Microsoft Windows 8 pro.
[Tagged as Positive]
```

The following topics are covered in the collection: Amazon, Apple, Facebook, Google, iPad, Lumia, Microsoft, Obama, Romney, Samsung, Twitter.

In this project, we combine both dataset and pick out 15% as evaluation data 15% as test data and the remaining 70% as training data.

Tweets in the dataset are imperfect. A small amount of tweets are mislabelled due to the limitation of our predefined rules. For instance, "Twitter on a iPad is horrible." is labeled with negative for topic "iPad". Also, "Youtube for iPad is awesome." is labeled with positive for topic "iPad".

What's more, some of the tweets are hard to label. For example, "Why is twitter so addicting?" can be interpreted as either positive or negative without context. Also, "Instead of updating your #Twitter why don't you update your life.". The implied sentimental meaning for twitter can be positive for its strong attraction or negative for its wasting too much time.

But overall, the number of tweets affected by those problem is relatively small and is ignored.

2.2 Parsing

Stanford parser is used for constituency parsing and dependency parsing. Ark tagger(Section[3.2.7]) is used for POS tagging.

2.3 Data Level Parallelism

During the model training step, training data (bulks of tweets) are sent to the feature extractor to generate a feature vectors. Extracting features are independent for every tweet in this step. So we are parallelizing the feature extraction step.

We utilize Java built-in thread pool model to do the parallelism. In this project, maximum running thread number equals to the core number of the machine.

On a machine (4 16-Core 2.1GHz AMD Opteron 6272 with 256GB memory), it achieves 7 times performance boosting comparing to the sequential version.

2.4 Preprocessing

Due to the natures of tweets, data collection needs to be preprocessed for next stage processing. The following are the rules for preprocessing:

1. Convert tweet to lowercase.
2. Remove URLs [] () and hashtag(#).
3. Remove the "rt" (Re-tweet) keyword.
4. Expand word abbreviation ("can't", "don't", "u", "r" etc). Example: [don't] - [do][not].
5. Remove all non-English characters (Japanese, Chinese etc).
6. Merge duplicated signs. (!!! - !)
7. Mapping quotation to NN. For example:"UNCLE SAM" launched a new policy last week. going to be translated to:"Uncle_Sam"-NN launched a new policy last week.
8. remove @ sign: If the following word starts with @ sign is the topic, simply remove the "@", otherwise, delete the whole word.
9. Trim invalid start and stop words. Example: "— I love pad. #@#" - I love pad.
10. Extract emoji symbol. Example: :(what's the solution? - ":(" symbol is extracted as negative token.

Tweet is pruned for next step processing:

Listing 2: preprocessing for tweet

```
Before:
RT :( what s the solution???? Nokia! "@NusibaT
    : @elle\_batool
\#Samsung is crap too mine keeps blocking calls
    from
coming through" &gt;&gt; http://t.co/IkKV9mxy

After:
[what][is][the][solution][?][Nokia][!][Samsung][
    is][crap][too][mine][keeps][blocking][calls
    ][from][coming][through]
```

2.5 Maximum Entropy Classifier

We use supervised learning algorithm with maximum entropy classifier. Maximum Entropy Model tries to find a prediction as the following:

$$prediction(x^i, w) = \arg\max w^T f_i(y)$$

The maximum entropy model we build would be stored into a file, from which the classifier could load it when performing the prediction. Thus, there is no need to build the model every time. Moreover, the features extracted can be either string or real number.

3 Experiments and Discussion

3.1 Lexical Model

At the first stage of our model, we built a Max Entropy model using Lexical Features. To tune our model, two aspects of parameters of the model were adjusted:

1. Different features
2. Different classifiers

3.1.1 Features

Following features are used in our lexical model. Here is a sample tweet and all following features would be illustrated based on this tweet.

Topic: ipad Tweet: @BereniceMoore - Why?! Its nothing bad. I'm gonna FB inbox you though, this twitter for ipad is crap

Topic replace First of all, we replace all topic word to tag TOPIC.

Emoticons Emoticons are extracted based on emoticon list.

Listing 3: emoticon

```
posEmoji_normalized=0.0, negEmoji_normalized=0.0
```

Lexical We defined words that tend to be positive or negative as sentiment words. Lists of lexicals that are tagged as positive/negative (Harvard Inquirer) are used to identify corresponding sentiment word in tweets. After identifying each sentiment word, we replace it as its sentiment tag when extract Bi-gram and Tri-gram features.

Sentiment word normalized frequency The normalized frequency of sentiment word is used to indicate which kind of sentiment word (positive/negative) appears more in current tweet. The feature is calculated as following:

$$POSNORMALIZED = \frac{c(POSITIVE)}{c(NEGATIVE)+c(POSITIVE)}$$
$$NEGNORMALIZED = \frac{c(NEGATIVE)}{c(NEGATIVE)+c(POSITIVE)}$$

E.g.
pos_count_normalized=0.0, neg_count_normalized=1.0

N-gram Unigram, Bigram and Trigram features are used.
E.g.
unigram_why
bigram_nothing_NEGATIVE
trigram_TOPIC_is_NEGATIVE

Negation A negation particle is attached to a word precedes it and follows it([Pak and Paroubek(2010)]). For example, a tweet "I do not like fish" will form three bigrams: "I do+not", "do+not like", "not+like fish". Such a procedure improves the accuracy of the classification since the negation plays a special role in opinion and sentiment expression ([Wilson et al.(2009)Wilson, Wiebe, and Hoffmann]). In English, we used negative particles "no" and "not".
E.g.
bigram_nothing_NEGATIVE
bigram_is_NEGATIVE

Distance features Position information finds its way into features from time to time([Pang and Lee(2008)]). We also calculate features to indicate the distance relationship between a sentiment word and the topic word:

neg_nearer , pos_nearer By calculating the nearest distance between the topic word and Negative, Positive word in the tweets, it is easy to find out which kind of sentiment word is nearer to topic word, based on basic sentence structures, the nearer word is more likely to modify the topic word.

neg_nearby , pos_nearby

At the same time, we can find out is there a sentiment word nearby the topic word. We judge a sentiment word is nearby a topic if its distance with topic word is less or equal than 5 words. Nearby sentiment words are more likely to modify the topic word.
E.g.
pos_nearby=0
neg_nearer=1
pos_nearer=0
neg_nearby=1
Following table shows the feature tuning process.

Features	F-Score(Pos)	F-Score(Neg)	F-Score(Neu)
word frequency	12.85	51.65	39.07
Unigram	77.85	74.81	74.83
Bigram	78.87	77.58	76.61
Trigram	78.13	78.28	74.66
Negation	79.81	78.48	76.69
Distance	80.54	79.07	78.01

3.1.2 Classifiers

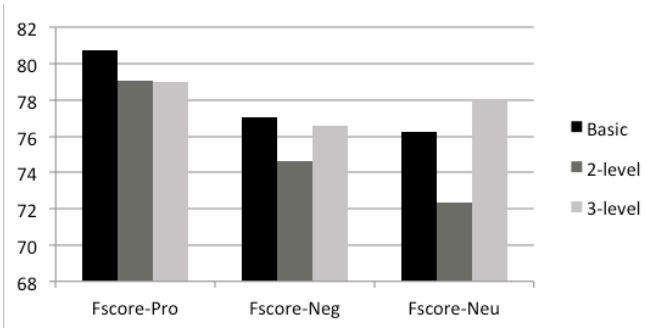
We construct 3 different classifiers for model tuning.

Basic Classifier This classifier is a Tri-nary classifier that would tag each tweet with basic sentiment categories: Positive, Negative and Neutral.
E.g. Neutral [0.0085] negative [0.0077] positive [0.9839] In above example, the tweet would be tagged as positive, since it has the highest probability.

2-level Binary Classifier The 2-level binary classifier has two binary sub-classifiers:
Neutral / Non-Neutral classifier: This classifier is trained with data tagged as only two categories, neutral and non-neutral (positive or negative) and tags tweet as these two tags.
Negative / Positive: This classifier is trained with data tagged as Negative or Positive (neutral tweets are eliminated) and tag tweet in the same way.
We assume that each binary classifier should be more confident than the basic tri-nary classifier. Therefore, we would use the neutral/non-neutral classifier to tag a tweet first, if the tweet is tagged as non-neutral, then we would back off and use the negative/positive classifier to decide the tweets final sentiment.

3-level Binary Classifier This classifier works in the same way as the 2-level classifier, but formed slightly differently with 3 binary classifiers:
Neutral / Non-Neutral classifier
Positive / Non-Positive classifier
Negative / Non-Negative classifier
During tagging, we tag each tweet with the Neutral / Non-Neutral classifier first, if tagged as non-neutral, we back-off to the Positive / Non-Positive classifier. If tagged as non-positive, then we back-off the last classifier and decide the final tag.
Following chart shows the comparison among 3 different classifiers.

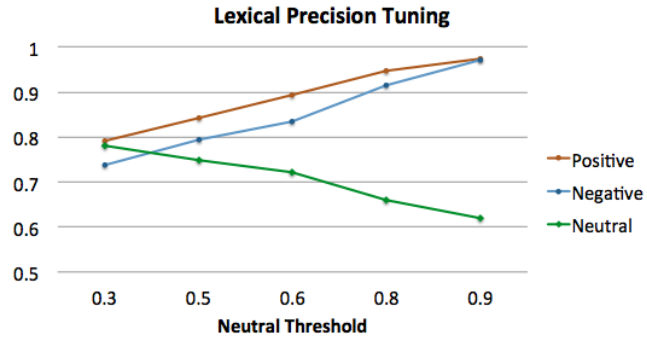
Figure 1: Classifier Comparison

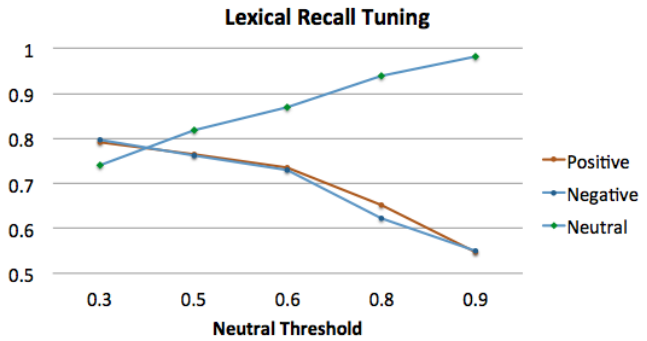


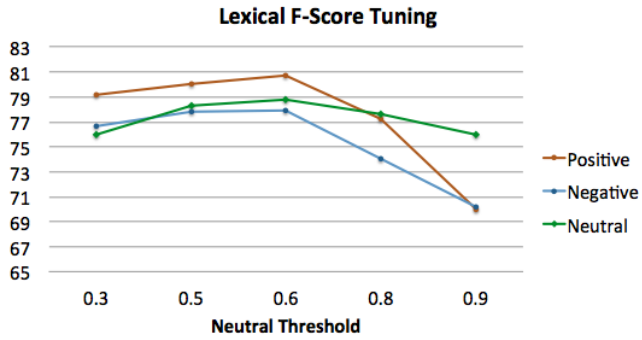
As is shown above, the multi-level classifier didn't improve too much compared with the basic classifier, therefore, basic lexical classifier would be used in following experiments.

3.1.3 Lexical Model Tuning

When the probablity of 3 categories are very close, well tag current tag as neutral. Therefore, we set a neutral threshold: when the probablity of the best outcome is lower than the neutral threshold, then it will be classified as neutral. Following charts show the result of tuning the threshold on validation data.







Based on above charts, we can see that the precision of neutral is decreasing while other two categories precision are increasing along with the growth of neutral threshold. Meanwhile recall has the opposite trend. That is because while increase the threshold, more tweets tend to be tagged as neutral, therefore neutral recall is increasing and precision is decreasing. The best result we achieved by tuning the threshold is as following(when $\lambda = 0.6$):

Best Result	Positive	Negative	Neutral
F-score	81.77	77.51	78.65
Recall	0.76	0.74	0.84
Precision	0.87	0.81	0.73

3.2 Grammatical Model

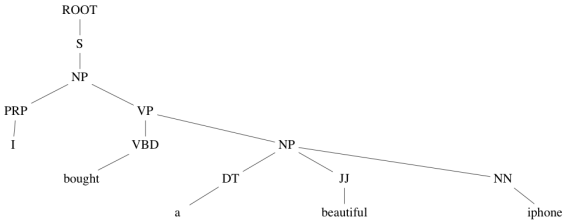
3.2.1 Inefficiency of Using Lexical Model

Lexical Model has its limitations when tweets contains multiple keywords that have contradictory meanings or complex sentence structures. Lexical Model recognize the number of positive words and negative words and their distance to the topic word. But it does not capture explicit relationships between words. On the other hand Grammatical Model captures relationships between words and therefore able to deal with such situation.

3.2.2 Parsing

Tweet is parsed using "stanford parser". To derive relationships/dependencies among words, the tweet should first be POS tagged and used to construct a constituency parsing tree. For the tweet "I bought a beautiful iphone", following parsing tree is constructed from stanford parser.

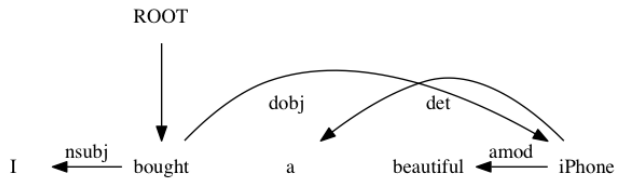
Figure 2: Constituency Parsing with Stanford Parser



3.2.3 Dependency Parsing

Dependency parsing shows the relationships between words. In the above tweet, we are interested relationships/dependencies that reveals sentimental information, in this case, its "beautiful-iphone". By providing a constituency parsing tree, a dependency parsing tree will be generated as shown in the following:

Figure 3: Dependency Parsing Tree



Listing 4: dependencies

```
[nsubj(bought-2, I-1), root(ROOT-0, bought-2), det(
    iphone-5, a-3), amod(iphone-5, beautiful-4),
    dobj(bought-2, iphone-5)]
```

amod(iphone-5,beautiful-4) shows the dependency between iphone and beautiful: beautiful is an adjective modifier to iphone. Following section discuss features captured from such dependency parsing tree.

3.2.4 Features From Dependency Parsing

For sentiment analysis, we focus on certain types of dependency between topic word and other word in the sentence.

1. Transitive verb and Topic relation: capture dobj(dominate object) dependency between verb and the topic word.
   given "I hate Microsoft" and topic "Microsoft", feature love_topic should be captured
2. Topic and Transitive verb relation: capture nsubj(noun subject) dependency between topic and verb word.
   given "Microsoft rocks!" and topic "Microsoft", feature topic_rocks should be captured
3. Adjective head is the topic, or say, Adjective directly modifies topic: capture amod(adjective modifier) dependency between adjective word and topic word.
   given "I bought a beautiful iPhone" and topic "iPhone", beautiful_topic is captured
4. Extract relative clause modifier: capture nsubj dependency between adjective word and topic word.
   given "I bought an iPhone, which is beautiful", beautiful_topic is captured
5. Extract Adjective+copula+topic: capture prep_xx dependency
   given "I am keen on iPad". keen_on_topic should be captured
6. Extract Topic+verb+adj: capture dep between a verb and topic and then capture advmod relationship
   given "iPhone runs smoothly comparing to android" and topic "iPhone", topic_run_smoothly is captured
7. Extract general dependency:
   given "Apple : Siri is amazing !" amazing_dep_topic is captured

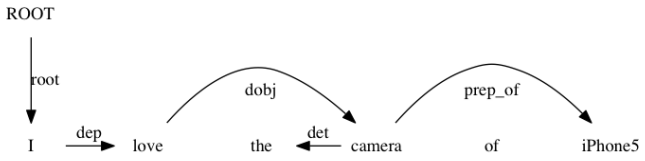
3.2.5 BFS Feature

Besides the dependencies and relationship direct towards the topic word, we also capture dependencies for tweets like:

"I love the camera of iPhone5"

In this case, camera has a positive sentiment, and is related with the topic word. as shown in following dependency parsing:

Figure 4: Dependency Parsing Tree



To capture features like this, we introduce a more fuzzy way to discover dependencies among words: Breadth First Search features. We start from certain types of words that may carry sentimental information, they are "JJ", "JJS", "JJR", "RB", "RBR", "RBS", "VB", "VBD", "VBN", "VBG", "VBP", "VBZ". Then we traverse to the topic word and record the path. Finally we concatenate the node on the path with the relation type. So we will get features like "I love working in google", BFS_love_xcomp_working_prep_in_topic is captured "I love the camera of iPhone5", BFS_love_dobj_iphone_poss_topic is captured

To reduce the dimension and make the features more general words in the middle of the traversal path are replaced with its pos tag. So for tweet "I love the camera of iphone 5", BFS_love_dobj_[NN]_poss_topic is captured

3.2.6 Negation Handling

Negation of a word carries opposite meaning, from a dependency parsing tree of a tweet, negation can be tracked by capture "neg" dependency. To handle negation, we prefix the word with "neg_" whenever a negative relationship is captured. So for "I don't love iphone", neg_love_topic will be captured.

3.2.7 Enhance Tokenization and POS Tagging

One of the difficulty for analyzing Tweets is its informality. Following are two samples:

Listing 5: tweet with special tokens

```
Texas \#Rangers are in the World Series!   G o
    Rangers!!!!!!!!! http://fb.me/D2LsXBJx \#
    trangers
RT @eye\_ee\_duh\_Esq: LMBO! This man filed an
    EMERGENCY Motion for Continuance on account of
    the Rangers game tonight! << Wow lmao
```

We came up manually pruning rules(Section[2.4]), but to further improve the pruning procedure for parsing, we need a specific POS tagger that deals with those special tokens in tweets, like hashtags, user mentioning, retweet mark etc.

The Ark group in Carnegie Mellon University provides fast and robust Java-based tokenizer and part-of-speech tagger for Twitter, its training data of manually labeled POS annotated tweets. Besides better tokenization, it provides better POS tagging. We use the tagger in following aspects.

1. Enhance POS tagging

   By default, stanford parser will tag each word with a pos tag which doesnt work well for tweets. So we used ark tagger to tag the sentence first and then pass it to the stanford parser along with their POS tags

2. Tweet specific pruning

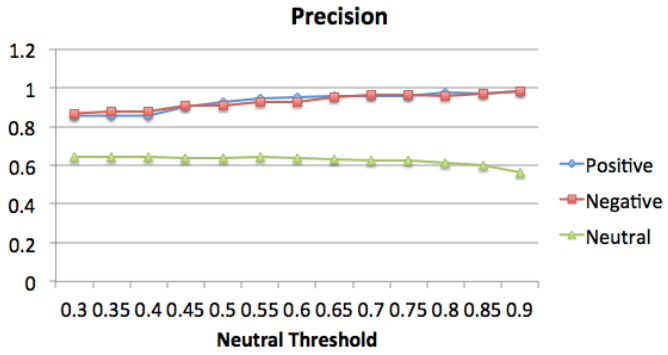
   Since Ark tagger tags Garbage Word, URL, emoticon, RT, we use those information to prune the tweets for better parsing result. We start from the first word and prune the sentence until we reach a word that is not Garbage Word, URL, RT. Following figure shows how a tweet is tagged and pruned.

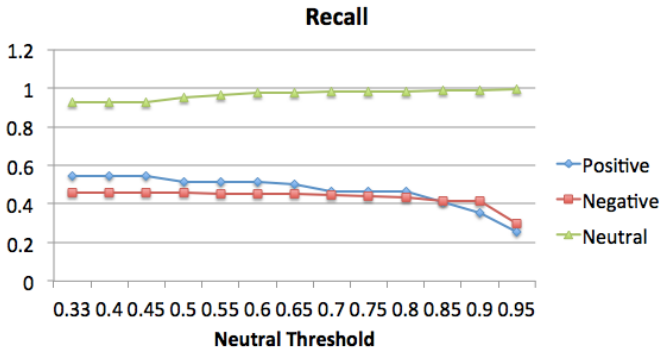
Figure 5: Ark Tagger For Pruning

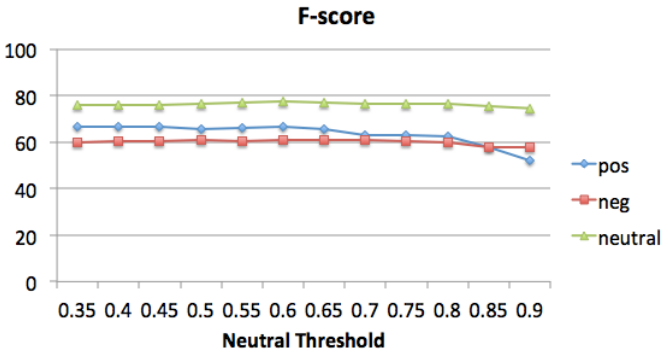
```
INFO : tweet:[apple]RT @eye_ee_duh_Esq: LMBO! This man filed an EMERGENCY
 Motion for Continuance on account of the Rangers game tonight! « Wow lmao
INFO : using ark tagger
INFO : ~  |  RT
INFO : @  |  @eye_ee_duh_Esq
INFO : ~  |  :
INFO : !  |  LMBO
INFO : ,  |  !
INFO : D  |  This
INFO : N  |  man
INFO : V  |  filed
INFO : D  |  an
INFO : N  |  EMERGENCY
INFO : N  |  Motion
INFO : P  |  for
INFO : N  |  Continuance
INFO : P  |  on
INFO : N  |  account
INFO : P  |  of
INFO : D  |  the
INFO : ^  |  Rangers
INFO : N  |  game
INFO : N  |  tonight
INFO : ,  |  !
INFO : ~  |  «
INFO : !  |  Wow
INFO : !  |  lmao
INFO : after preprocess[This->DT, man->NN, filed->VB, an->DT, EMERGENCY->NN,
 Motion->NN, for, Continuance->NN, on, account->NN, of, the->DT,
 Rangers->NNP, game->NN, tonight->NN, ., «, ,, ,]
```

3.2.8 Neutral Threshold Tuning

As discussed in Section[3.1.3], we use Neutral Threshold to finer tuning the classifier, when the Neutral Threshold is higher, the classifier tends to classify the word to neutral instead of positive or neutral. Following figures shows the result of classification with neutral threshold from 0.33 to 1.0







When neutral threshold=0.55, we have the best result:

Best Result	Precision	Recall	F-score
Positive	0.927	0.513	66.01
Negative	0.907	0.453	60.47
Neutral	0.639	0.962	76.8

Comparing to Lexical Model, grammatical model has a high precision and low recall. In Section[3.3] and Section[3.4], we discuss how to take advantage from both grammatical and lexical model to increase the precision and recall.

3.3 Backoff Model

Since grammatical model provides higher precision, we use the grammatical model to classify the tweet first, and according to some rules, if the prediction is not confident then we backoff to lexical model. We build the back off model as follows:

1. Try to classify the tweets with grammatical model

2. if 1) the features extracted is empty(this happens when a tweet can not be parsed by stanford parser)

or

2) if the probablity of the prediction is less than a backoff threshold then back off to a lexical model

Different backoff threshold are tuned as shown in the following figure

Figure 6: Backoff Tunning

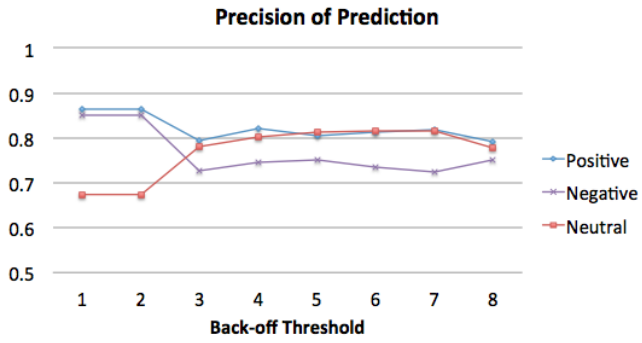


Figure 7: Backoff Tunning

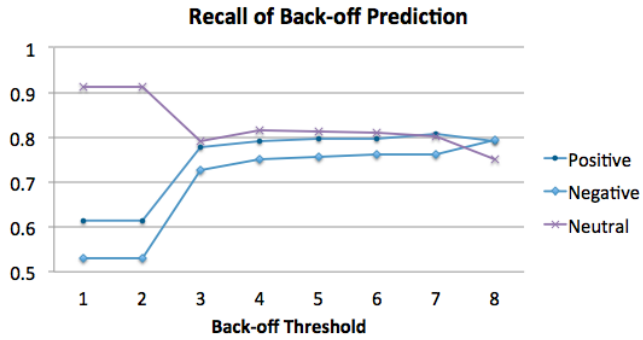
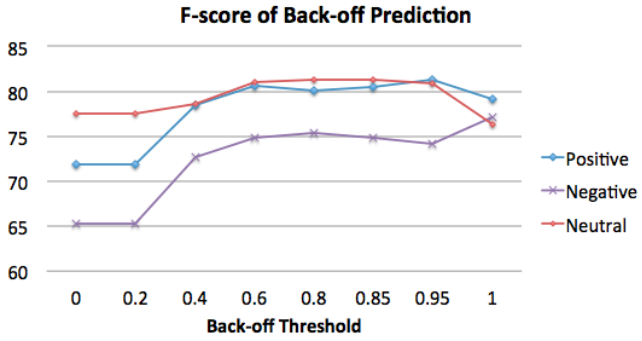


Figure 8: Backoff Tunning



When backoff threshold is set to 0.8 we get the following result:

Best Result	precision	recall	f-score
positive	80.5	79.7	80.1
negative	75.1	75.5	75.3
neutral	81.1	81.4	81.3

The backoff model outperforms lexical model and grammatical model.

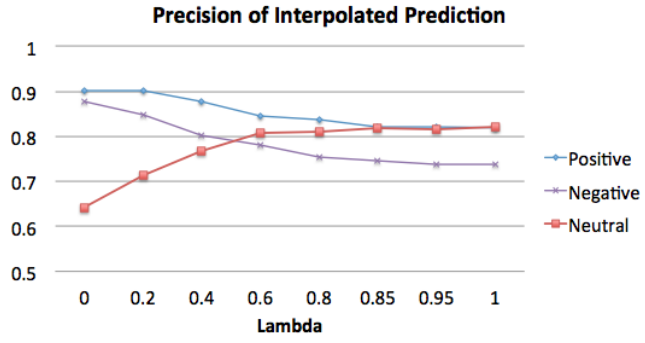
3.4 Interpolated Model

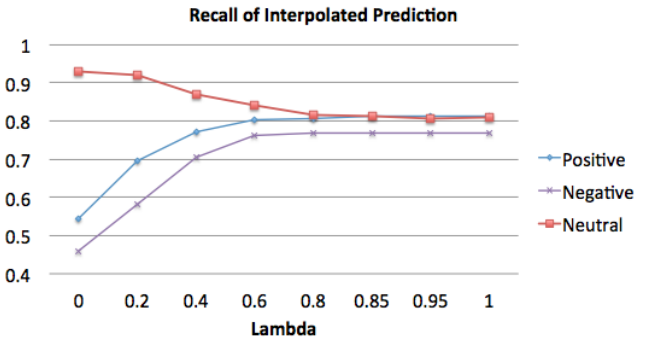
Another comprehensive model we built is an interpolated model. Using following formulas we combined the prediction result from both lexical and grammatical models.

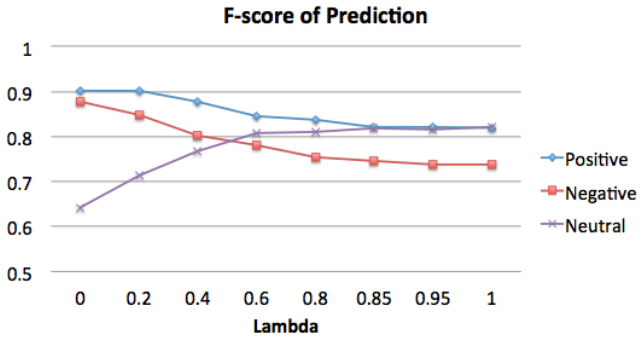
$$P(class) = \lambda P_{lexical}(class) + (1-\lambda) P_{grammatical}(class)$$
$$prediction = \arg\max_{sentiment} P(class)$$

By tuning the value of λ, we tuned the weight of lexical and grammatical model, the higher the λ is, the more probable is lexical model dominating the result. Therefore, the higher the precision should go while the recall should decrease.

Following charts show our tuning result.







Based on the tuning result, following table shows the best result we achieved: When $\lambda = 0.85$

Best Result	Positive	Negative	Neutral
F-score	82.4	76.4	81.7
Recall	0.812	0.767	0.811
Precision	0.821	0.745	0.818

4 Conclusions

Tweets are very different from news text and therefore require special treatment. Pruning noisy information is an effective way to get more formal sentences for analysis and parsing In Section[2.4] and Section[3.2.7], we used word level pruning techniques which reduces the amount of noisy words.

Parsing can be used to get more precise result by capturing relationship among words. But it fails when the tweet cannot be parsed correctly which happens very often. On the other hand, Lexical Models will always able to extract features from a sentence but fails when a tweet contains complex sentiment information. Therefore, a combined approach takes advantages from both lexical model and grammatical model and outperforms both models.

References

[Jiang et al.(2011)Jiang, Yu, Zhou, Liu, and Zhao] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao. Target-dependent twitter sentiment classification. Now Pub, 2011.

[Pak and Paroubek(2010)] A. Pak and P. Paroubek. Twitter based system: Using twitter for disambiguating sentiment ambiguous adjectives. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 436–439. Association for Computational Linguistics, 2010.

[Pang and Lee(2008)] B. Pang and L. Lee. Opinion mining and sentiment analysis. Now Pub, 2008.

[Wilson et al.(2009)Wilson, Wiebe, and Hoffmann] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational linguistics*, 35(3):399–433, 2009.