Sentiment Analysis on Movie Reviews with Feature Grammars

Razieh Mehri

This project shows how feature grammars are used to improve the sentiments analysis for movie reviews.

Introduction

In this project, we built a tool that improves the Simplest Sentiment Analysis in Python (SSAP) tool for rating movie reviews. In this report, first the three papers that our project implementation is inspired by will be introduced. Then, the movie review dataset used in the project is introduced. Afterward, the introduction of Simplest Sentiment Analysis in Python (SSAP) on movie review dataset will be given. Finally, after presenting our implementation, we will analyze and criticize our implementation.

Papers on Sentiment Analysis

There have been many researches on sentiment analysis in recent years. One of the research papers called **"**Mining Opinion in Comparative Sentences**"** exploring the sentiment analysis in product reviews seems very interesting. Authors expressed a very good point of view on comparative sentences. The paper inspired us for considering comparative adjectives for our implementation. Second paper is **"**Survey on the Role of Negation in Sentiment Analysis**".** The paper is very useful for modeling negations in sentiments analysis. We also inspired by some of the methods they used to detect the sentiment of phrases with negation. Third paper is "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews". It follows an interesting pipe line contains the extracting of phrases and estimating their sentiments. Preprocessing and detecting the phrases for sentiment estimation were the steps which took our attention in this paper.

Dataset

The dataset we used for our project and also running the baseline is Rotten Tomatoes Movie Reviews. The dataset consists of separate sentences. Most of the sentences are difficult to parse since they are not complete sentences or composed of different sentences. We chose the sentences which we can build grammar for them and also can show the capability of our implementation with.

SSAP

Simplest Sentiment Analysis with Python only looks for the polarities of the words exist in AFINN dictionary and adds them. Finally, the sentiment of the sentence is given by dividing the added polarities by the number of the words in the sentence. For example, while determining the

sentiment of sentence “He is not a bad boy.” using SSAP, it only considers the polarity of the word “bad” exist in the AFINN dictionary and will not consider the negation word. In our implementation, we will consider the negation word “not” which modifies the polarity of the word “bad”.

The results of the baseline on a few sentences we parsed during our implementation are presented later.

Implementation

In our implementation, we tried to take the advantages of the feature based grammars as much as possible.

Most of the sentences in dataset have punctuation marks such as parenthesis, brackets, commas, etc., so they should be removed before being parsed by our grammar.

The feature based grammar (SenGrammar.fcfg) used to implement our idea is not public but some parts are presented in this report.

In every sentence parsed by the feature grammar, we look for the children of it’s **S** node (sentence) in order to find out how to obtain the sentiment value of the sentence. Based on the type of sentence (MOOD Feature of sentence), we shall decide how to explore the tree and whether the MOOD of the sentence will have any impact on the whole sentiment value obtained for the sentence or not.

We tried two moods for sentences in the implementation: declarative(dec) and question (wh). Each of them will be explained in more details in the following:

```
S[MOOD=dec, AGR=?n, TENSE=?t, SENT=?ss] -> NP[AGR=?n, SENT=?nps] VP[AGR=?n, TENSE=?t, SENT=?vps] period

S[MOOD=dec, AGR=?n, TENSE=?t, SENT=?ss] -> RB[advtype='Comment', SENT=?ss] NP[AGR=?n, SENT=?nps] VP[AGR=?n, TENSE=?t, SENT=?vps] period

S[MOOD=wh, AGR=?n, TENSE=?t, SENT=?ss] -> w_word  VP[AGR=?n, TENSE=?t, SENT=?ss] quest_mark
```

For sentences in question mood, we shall decrease the value of the sentence after we figure out it is in question mood.

```
S[MOOD=wh, AGR=?n, TENSE=?t, SENT=?ss] -> w_word  VP[AGR=?n, TENSE=?t, SENT=?ss] quest_mark
```

For example,

“who needs love like this ?”

shows that the reviewer is surprised and don't seem to enjoy the movie. Reviewing most of the sentences in question mood, we found out that question mood often indicates a negative review. Therefore, the value of the sentiment we obtained by parsing the rest of the sentence should be decreased after we found out the mood of sentence to be question (MOOD='wh').

For this specific example, the polarity of the sentence will be flipped (switch from positive to negative), From 2.04 to -2.04. In the case where the polarity of the sentence is neutral in first place, we decrease it by 2 in order to make the sentence's polarity negative.

To know how to get the polarity of the sentence at first place, this will be discussed later.

We also tried to write/modify our feature grammar to be more flexible to sentiment analysis. Therefore, declarative sentences that our grammar can parse are in two types: First groups are those declarative sentences start with comment adverbs such as: gladly, happily, unfortunately, etc. and complete with NP and VP. The second group is normal declarative sentences which are composed of NP and VP. This grouping helps us to detect the sentiments of declarative sentences with different structures much easier.

Compared to the second group, the first group of declarative sentences; in other words, adverbial sentences shows the reviewers' point of view instantly. For example, a sentence from review dataset says:

"unfortunately the story and the actors are served with a hack script ."

We defined the following productions for our grammar to parse this sentence:

```
S[MOOD=dec, AGR=?n, TENSE=?t, SENT=?ss] -> RB[advtype='Comment', SENT=?ss] NP[AGR=?n, SENT=?nps] VP[AGR=?n, TENSE=?t, SENT=?vps] period

RB[advtype='Comment', SENT='neg']-> "unfortunately"

RB[advtype='Comment', SENT='pos']-> "gladly"
```

Grammar rules for RB(adverb) shows how the advtype(adverb type) and SENT(sentiment) features of adverb can help us to make a decision in order to change the polarity of the whole sentence(especially if the polarity of the NP+VP is zero). As it shows, the SENT feature of sentence is similar to the SENT feature of the comment adverb.

Depend on the positivity, negativity and neutrality of the rest of the sentence (NP+VP), we shall decide how the comment adverb can change the polarity value. But In our example where the sentiment of NP+VP is neutral, we decrease their sentiment by 3. Therefore, the sentiment of sentence changes from 0 to -0.83.

The parse tree for the above sentence is:

(SG[]

```
(S[AGR='Third_Plural', MOOD='dec', SENT='neg', TENSE='T_Pres_Prog']
 (RB[SENT='neg', advtype='Comment'] unfortunately)
 (NP[AGR=?n, SENT=?nps]
  (NP[AGR='Third_Single']
   (DETERMINER[AGR='Third_Single']
    (article[AGR='Third_Single']
     (word_the[AGR='Third_Single'] the)))
   (NOUN[AGR='Third_Single']
    (count_noun[AGR='Third_Single'] story)))
  (aux_and[] and)
  (NP[AGR='Third_Plural']
   (DETERMINER[AGR='Third_Plural']
    (article[AGR='Third_Plural']
     (word_the[AGR='Third_Plural'] the)))
   (NOUN[AGR='Third_Plural']
    (count_noun[AGR='Third_Plural'] actors))))
 (VP[AGR='Third_Plural', GROUP='G_Passive', SENT=?vps, TENSE='T_Pres_Prog']
  (aux_be[AGR='Third_Plural', FORM='F_Simp_Pres'] are)
  (i_or_t_verb[AGR='Third_Plural', FORM='F_Past_Princ', SENT=?iotvs]
   (trans_verb[AGR='Third_Plural', FORM='F_Past_Princ'] served))
  (VERB_COMP[AGR=?n]
   (PP[AGR=?n]
    (AUX_PP[] (aux_with[] with))
    (NP[AGR=?n, SENT=?nps]
     (NP[AGR='Third_Single']
      (DETERMINER[AGR='Third_Single']
       (article[AGR='Third_Single'] a))
      (NOUN[AGR='Third_Single']
       (count_noun[AGR='Third_Single'] hack)))
     (NP[AGR='Third_Single', SENT=?nps]
      (NOUN[AGR='Third_Single']
       (count_noun[AGR='Third_Single'] script)))))))
 (period[] .)))
```

In the second group of declarative sentences (normal sentences composed of NP+VP), we should look for the sentiment feature of NP and VP to find out how to combine them in order to get the sentiment of the sentence.

S[MOOD=dec, AGR=?n, TENSE=?t, SENT=?ss] -> NP[AGR=?n, SENT=?nps] VP[AGR=?n, TENSE=?t, SENT=?vps] period

The following table shows the combination rules for NP and VP polarities.

In the following table, SENT(NP) stands for the SENT feature of NP which could be positive("pos"), negative("neg") or neutral(0). We didn't define neutral value in our implementation but when it's empty or [], we know it is neutral.

IF	THEN
SENT(NP)=0 & SENT(VP)="neg" or "pos"	SENT(S)=SENT(VP)
SENT(NP)= "neg" or "pos" & SENT(VP)=0	SENT(S)=SENT(NP)
SENT(NP)="pos" & SENT(VP)="neg"	SENT(S)=SENT(NP) + 2× SENT(VP)
SENT(NP)="pos" & SENT(VP)="pos"	SENT(S)=SENT(NP) + SENT(VP)
SENT(NP)="neg" & SENT(VP)="neg"	SENT(S)=SENT(NP) + SENT(VP)

SENT(NP)=”neg” & SENT(VP)=”pos”	SENT(S)=2×SENT(NP) + SENT(VP)

Therefore, the sentiment feature helps us to obtain the polarity of the whole sentence using the above rules.

Now it’s time to find the polarity of NP and VP. NP also exits inside Verb Complement (VERB_COMP) of the VP. Therefore, we focus on how to get the sentiment of VP, i.e., NP inside VP.

VP is composed of a Verb and a VERB_COMP (Verb complement). We shall look for the leaves, i.e., their left or right siblings of the specific subtree in order to find the polarity of the NP, i.e., VERB_COMP.

Verb can be negative (explicit and implicit) or neutral or positive.

While the positive verbs such as *boost* will multiply the polarity of the Verb Complement, the negative verb (implicit) such as *avoid*, *prevent*, etc will divide the polarity of the Verb Complement.

For example, for

“a work that lacks both a purpose and a strong pulse .”

we added the following productions to indicate the negativity of the verb :

VERB[AGR=?n, TENSE=T_Simp_Pres, SENT=?vbs] -> trans_verb[AGR=?n, FORM=F_Simp_Pres, SENT=?vbs]

trans_verb[AGR=Third_Single, FORM=F_Simp_Pres, SENT='neg'] -> "lacks"

This VERB(with SENT=’neg’) changes the positivity of VERB_COMP.

Based on the sentiment of the verb and the sentiment of the Verb Complement, we will decide how to obtain the polarity according to the following table.

IF	**THEN**
SENT(Verb)=0 & SENT(VERB_COMP)= ”neg” or “pos”	SENT(VP)=SENT(VERB_COMP)
SENT(Verb)=”neg” & SENT(VERB_COMP)= ”neg” or “pos”	SENT(VP)= (-1) × SENT(VERB_COMP)
SENT(Verb)=”pos” & SENT(VERB_COMP)= ”pos”or”neg”	SENT(VP)=2 × SENT(VERB_COMP)

SENT(Verb)=”pos” or “neg” SENT(VERB_COMP)=0	SENT(VP)=SENT(VERB_COMP)

In the above example we apply the second rule from table and flip the sentiment of the Verb Complement (switch from positive to negative); therefore, the polarity of sentence changes from 0.58 to -0.58

The parse tree for the above sentence is:

```
(SG[]
 (S[AGR='Third_Single', MOOD='dec', SENT=?ss, TENSE='T_Simp_Pres']
  (NP[AGR='Third_Single']
   (NP[AGR='Third_Single']
    (DETERMINER[AGR='Third_Single']
     (article[AGR='Third_Single'] a))
    (NOUN[AGR='Third_Single']
     (count_noun[AGR='Third_Single'] work)))
   (rel_pronoun[AGR='Third_Single'] that))
  (VP[AGR='Third_Single', GROUP='G_Active', SENT=?vps, TENSE='T_Simp_Pres']
   (VERB[AGR='Third_Single', SENT='neg', TENSE='T_Simp_Pres']
    (i_or_t_verb[AGR='Third_Single', FORM='F_Simp_Pres', SENT='neg']
     (trans_verb[AGR='Third_Single', FORM='F_Simp_Pres', SENT='neg']
      lacks)))
   (VERB_COMP[AGR='Third_Single', SENT=?vcs]
    (NP[AGR='Third_Single']
     (NP[AGR='Third_Single', SENT=?nps]
      (quantifier[AGR='Third_Plural'] both)
      (NP[AGR='Third_Single']
       (DETERMINER[AGR='Third_Single']
        (article[AGR='Third_Single'] a))
       (NOUN[AGR='Third_Single']
        (count_noun[AGR='Third_Single'] purpose))))
     (PP[AGR='Third_Single']
      (AUX_PP[] (aux_and[] and))
      (NP[AGR='Third_Single', SENT='pos']
       (DETERMINER[AGR='Third_Single']
        (article[AGR='Third_Single'] a))
       (JJ[SENT='pos'] strong)
       (NOUN[AGR='Third_Single']
        (count_noun[AGR='Third_Single'] pulse)))))))
  (period[] .)))
```

Comparative and superlative adjectives play important roles in sentiment analysis. In movie review sentences, we only focus on superlative adjective phrases. The reason is that we’re not reviewing items/products to care about comparative adjective phrases. The review sentences only regard to a single movie and usually there is not any comparison between two movies inside the sentences. Therefore, we add the following rules to our grammar for superlative:

ADJP[SENT='pos'] -> quantifierComparative[SENT='pos'] JJ[SENT='pos']

ADJP[SENT='neg'] -> quantifierComparative[SENT='pos'] JJ[SENT='neg']

ADJP[SENT='neg'] -> quantifierComparative[SENT='neg'] JJ[SENT='pos']

ADJP[SENT='pos'] -> quantifierComparative[SENT='neg'] JJ[SENT='neg']

quantifierComparative[qtype='superlative ', SENT='pos'] -> "most"

quantifierComparative[qtype='superlative ', SENT='neg'] -> "least"

Based on the positivity and negativity of both quantifier (most, least) and adjective we decide how to get the sentiment of adjective phrase according to the table:

IF	THEN
SENT(quantifierComparative)=”pos” & SENT(JJ)=”pos” or “neg”	SENT(ADJP)= 2 × SENT(JJ)
SENT(quantifierComparative)=”neg” & SENT(JJ)=”pos” or “neg”	SENT(ADJP)= (-2) × SENT(JJ)

Therefore, in sentence:

“it is one of the most honest films ever made about hollywood .”

the polarity will change from 0.55 to 1.11 after applying the rule.

The parse tree for that specific sentence is:

```
(SG[]
 (S[AGR='Third_Single', MOOD='dec', SENT=?ss, TENSE='T_Simp_Pres']
  (NP[AGR='Third_Single', SENT=?nps]
   (NOUN[AGR='Third_Single'] (pronoun[AGR='Third_Single'] it)))
  (VP[AGR='Third_Single', GROUP='G_Active', SENT=?vps, TENSE='T_Simp_Pres']
   (VERB[AGR='Third_Single', SENT=?vbs, TENSE='T_Simp_Pres']
    (i_or_t_verb[AGR='Third_Single', FORM='F_Simp_Pres', SENT=?iotvs]
     (trans_verb[AGR='Third_Single', FORM='F_Simp_Pres'] is)))
   (VERB_COMP[AGR='Third_Plural', SENT='pos']
    (NP[AGR='Third_Plural', SENT='pos']
     (cardinal[AGR='Third_Single'] one)
     (aux_of[] of)
     (NP[AGR='Third_Plural', SENT='pos']
      (DETERMINER[AGR='Third_Single']
       (article[AGR='Third_Single']
        (word_the[AGR='Third_Single'] the)))
      (ADJP[SENT='pos']
       (quantifierComparative[SENT='pos', qtype='superlative ']
        most)
       (JJ[SENT='pos'] honest))
      (NP[AGR='Third_Plural', SENT=?nps]
       (NOUN[AGR='Third_Plural']
        (count_noun[AGR='Third_Plural'] films)))
      (ADVP[AGR=?n]
       (AVP[] (DUR_ADV[] ever))
       (trans_verb[AGR='First_Single', FORM='F_Simp_Past']
        made))
```

```
      (PP[AGR='Third_Single']
        (word_about[] about)
        (NP[AGR='Third_Single', SENT=?nps]
          (NOUN[AGR='Third_Single']
            (proper_noun[AGR='Third_Single'] hollywood))))))))
  (period[] .)))
```

Many of the phrases found in the movie review sentences are in the form of "ADJP but ADJP"or "JJ but JJ". We implemented the second one which is simpler. We decided to add an extra rule to our grammar for declarative sentences.

S[MOOD=dec, AGR=?n, TENSE=?t, SENT=?ss] -> ADJP[AGR=?n, SENT=?ss] period

ADJP -> AJP_MOD CC AJP_MOD
ADJP[SENT='neg'] -> JJ[SENT='pos'] CC JJ[SENT='neg']
ADJP[SENT='pos'] -> JJ[SENT='neg'] CC JJ[SENT='pos']
ADJP[SENT=?jjs] -> JJ[SENT=?jjs] CC AJP_MOD
ADJP[SENT=?jjs] -> AJP_MOD CC JJ[SENT=?jjs]

Since the main clause is the one comes after but we will obtain the sentiment based on the below table:

IF	THEN
SENT(JJ1)="pos" or "neg" & SENT(JJ2)="pos" or "neg"	SENT(S)=SENT(JJ1) + 2 × SENT(JJ2)
SENT(JJ2)="pos" or "neg" & SENT(JJ2)=0	SENT(S)=SENT(JJ1)/2
SENT(JJ1)=0 & SENT(JJ2)="pos" or "neg"	SENT(S)= 2 × SENT(JJ2)

Two examples are:

"earnest but heavy-handed ."

Where the polarity changes from 1.00 to 0.50 and

"decent but dull ."

Where the polarity changes from -1.00 to -2.00 .

The parse tree for the first phrase is:

```
(SG[]
 (S[AGR=?n, MOOD='dec', SENT='pos', TENSE=?t]
  (ADJP[SENT='pos']
   (JJ[SENT='pos'] earnest)
   (CC[] but)
   (AJP_MOD[] (adj_mod[] heavyhanded)))
  (period[] .)))
```

This rule can be expanded for more complex adjective phrases but due to the limited time we only implemented for two adjective connected with conjunction *but*.

Many of the movie names have words with polarity values can be found in AFINN dictionary. Most of the movie names in movie review sentences are inside quotation which can easily be detected. We added them as movie names to the proper names. This gives us more accurate sentiment of a sentence.

For example, in sentence:

"the draw [for " big bad love "] is a solid performance by arliss howard ."

The sentiment is changed from 0.80 to 0.53 after detecting "big bad love" as a movie name.

When adverbs and adjectives come together, usually the adverbs are intensifiers and they can decrease or increase the polarity of adjectives based on their types: Strong intensifier (such as very, extremely), Weak intensifier(such as barely, slightly), Minimizer(such a hardly) or Compromiser(such as rather). In out grammar we only cover strong intensifiers. Therefore, when find an adjective in the tree leaves we look for left sibling of that node and if it is intensifier adverb we will change the polarity of the adjective based on the type of the adverb.

The following productions added to the grammar:

```
AJP[SENT='neg'] -> RB[advtype='Negative', SENT='neg'] RB[advtype='IntensifierStrong',
SENT='pos'] JJ[SENT='pos']
###INTENSIFIER ADVERBAL PHRASES###
RB[advtype='IntensifierStrong', SENT='pos']-> "very"
```

Further, we can have negative adverb comes before adjective phrase and can flip the sentiment value of the adjective phrase.

```
###NOT,NEVER ADVERBAL PHRASES###
RB[advtype='Negative', SENT='neg']-> "not"
```

In the sentence we chose from movie data set:

"on its own , it's not very interesting ."

We have negative adverb comes before string Intensifier companies an adjective. Since the

intensifier is strong and positive it multiplies the sentiment of positive adjective comes after it twice.

SENT(“Very interesting”)= 2 × SENT(“interesting”)

Then we check for any negation adverb comes before the adjective phrase and should flip (switch from positive to negative) the sentiment of the adjective phrase.
Since the adjective phrase has intensifier in it, we should reconsider flipping it and instead we only decrease the sentiment of adjective phrase by 3. The reason for doing this is we know that “*not very interesting*” is not as bad as “*not interesting*”.
Therefore, instead of 0.67, we got 0.33.

The parse tree for this example:

```
(SG[]
 (S[AGR='Third_Single', MOOD='dec', SENT=?ss, TENSE='T_Simp_Pres']
  (NP[AGR='Third_Single']
   (AJP[]
    (PP[AGR='Third_Plural']
     (AUX_PP[] (aux_on[] on))
     (NP[AGR='Third_Plural', SENT=?nps]
      (NOUN[AGR='Third_Plural']
       (pronoun[AGR='Third_Plural'] its))))
    (VERB_COMP[AGR='Third_Plural', SENT=?vcs]
     (NP[AGR='Third_Plural', SENT=?nps]
      (NOUN[AGR='Third_Plural']
       (mass_noun[AGR='Third_Plural'] own)))))
   (NOUN[AGR='Third_Single'] (pronoun[AGR='Third_Single'] it)))
  (VP[AGR='Third_Single', GROUP='G_Active', SENT=?vps, TENSE='T_Simp_Pres']
   (VERB[AGR='Third_Single', SENT=?vbs, TENSE='T_Simp_Pres']
    (i_or_t_verb[AGR='Third_Single', FORM='F_Simp_Pres', SENT=?iotvs]
     (trans_verb[AGR='Third_Single', FORM='F_Simp_Pres'] is)))
   (VERB_COMP[AGR=?n, SENT='neg']
    (AJP[SENT='neg']
     (RB[SENT='neg', advtype='Negative'] not)
     (RB[SENT='pos', advtype='IntensifierStrong'] very)
     (JJ[SENT='pos'] interesting))))
  (period[] .)))
```

English grammar also has some adjectives as intensifiers. If the word after those adjectives is noun and the noun has polarity inside AFINN, we will increase the polarity of the Noun.

JJ[jjtype='intensifier',SENT='pos'] -> "absolute"

Since the type of adjective is intensifier, the polarity of the adjective increaases twice. As an example, I parsed the following sentence and instead of 1.13, I got 2.27.

“the performances are an absolute joy .”

Results and Critique of Our Implementation

The following table shows the results of our implementation and SSAP on a few sentences of the movie review dataset plus the polarity classification (NEG and POS) given by the dataset provider.

SSAP		Our Implementation Sentiment		Dataset Sentiment (rt-polarity)
2.04	POS	-2.04	NEG	NEG
0	NEUT	-0.83	NEG	NEG
1.13	POS	2.27	POS	POS
0.67	POS	0.33	POS	POS
0.58	POS	-0.58	NEG	NEG
0.55	POS	1.11	POS	POS
0.80	POS	0.53	POS	POS
1	POS	0.50	POS	POS
-1.00	NEG	-2.00	NEG	NEG

Compared to SSAP, our implementation detects the positivity and negativity of the sentences correctly (close to the polarity provided by rt-polarity classification).

Even tough there still exist many patterns to be added to the grammar, the implementation can easily be expanded to the sentences using similar rules to be parsed. Due to the limited time, we only could build a few productions for the grammar and develop our python code for them. Still our idea takes the most advantages of the feature grammars and by combining the features from both grammars (different types of adverbs, adjectives, sentences, etc) and sentiments (positive, negative, neutral), we were able to come up with an idea which seems to be working good but still needs works to build our complete sentiment tool.