PROJECT REPORT

1. Group Information

Group Name: Rio

Group Member Names:

- Shizhe Bao
- Zhenqiao He

2. Problem Description

Our project is an Automatic Rating System, which use star(s) to analysis a review.

3. Proposed Solution and Fully Implementation Details

3.1 Corpus

Collect 100 to 200 reviews and stars rating from yelp.com.

3.2 Baseline System (Naïve Strategy)

Look for certain keywords that are indicators of certain star(s) (from 1 star to 5 stars).

3.3 Improvement Strategy

3.3.1 Lexical Features

- **Word Tokens:** In baseline system, we focus on key words. By improvement strategy, we focus on all related words in a review.
  Example: "I was not happy." In this sentence, the word "happy" is a positive review, which may get higher star rating. "not happy", however, makes the review worth 2 stars rating even 1 star.
- **Lemmatization:** To reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.
  Example: The word "hated" may not appear in the training set, however, after lemmatization, the word "hate" could be used to compute the probability of star(s) of review.

3.3.2 Syntactic Features

- **POS Tagging:** POS tags could be used for identifying and treating differently the different meaning of polysemous words.
  Example: The word "loathing" has different tags. In "Fear and Loathing in Las Vegas", "loathing" is a noun, which may not affect the probability of star(s) of review.
- **Dependency Parser:** Tell relationship between two words in a sentence based on the dependency relation (as opposed to the constituency relation).

Example: For the sentence “I love this movie”, we can get the following results: nsubj(love-2, I-1), root(ROOT-0, love-2), det(movie-4, this-3) and dobj(love-2, movie-4), which may affect the probability of star(s) of review.

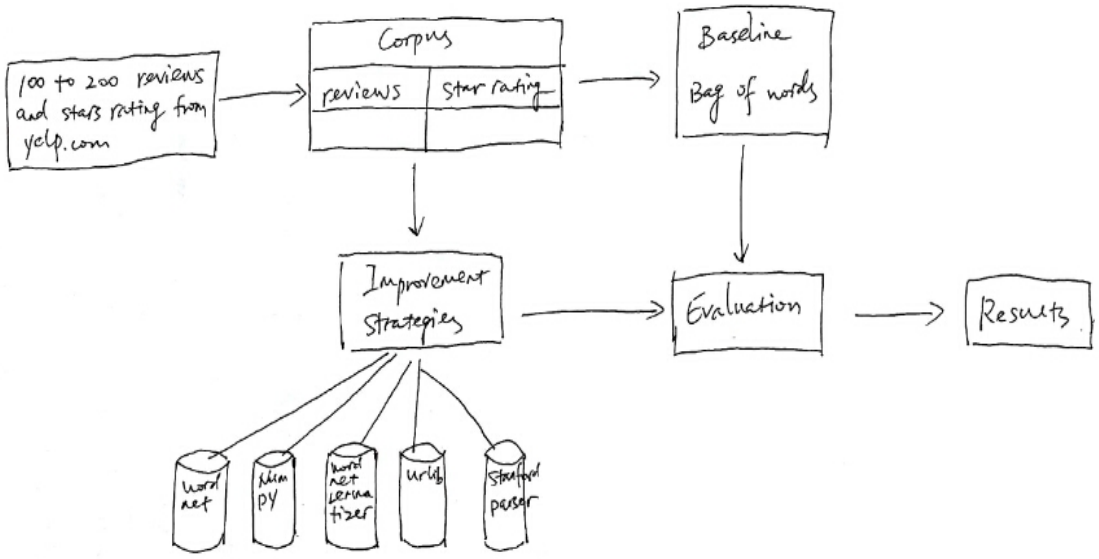
3.3.3 Semantic Features

- **LESK:** Based on the assumption that words in a given "neighborhood" (section of text) will tend to share a common topic.
  Example: The word “happy” has multiple meanings. We use LESK algorithm to find the correct meaning of “happy” in the sentence, which may affect the probability of star(s) of review.
- **Semantic Relations:** Tokens in a sentence have same concept and relationship.
  Example: Synonymy. The sentence “I love this movie” has the synonymy words “me like it film”, which may affect the probability of star(s) of review.

3.4 Programming Tools

Python 2.7, NLTK (wordnet, wordnetlemmatizer), Numpy, urlib, urlib2, re, The Stanford Parser

3.5 Architecture Diagram



4. Results

4.1 Naive Implementation

naive.py — UNREGISTERED

```
from url import Comments
import re
comments=Comments()
testlist=comments.testList
pattern=re.compile(r'[.,"\?!:-]')
set1=['rediculous','terrible','horrible']
set2=['disappoint','small']
set3=['popular','hot','thankfully']
set4=['good','love','enjoy']
set5=['awsome','gorgeous','delicious']
print "naiave result:"
def getStar(comment):
    words=comment.split(' ')
    for word in words:
        word=re.sub(pattern,'',word)
        if word in set1:
```

```
naiave result:
original: 4     naive: 4
original: 3     naive: 3
original: 3     naive: 4
original: 4     naive: 5
original: 4     naive: 4
original: 4     naive: 5
original: 3     naive: 3
original: 3     naive: 4
original: 2     naive: 3
original: 2     naive: 2
original: 3     naive: 3
original: 3     naive: 2
original: 2     naive: 4
original: 4     naive: 4
original: 4     naive: 2
original: 3     naive: 4
original: 4     naive: 4
original: 3     naive: 4
original: 4     naive: 4
original: 3     naive: 4
naive accuracy: 0.45
[Finished in 11.9s]
```

Line 7, Column 28 — Tab Size: 4 — Python

4.2 Word Token (Lexical Features)

separate.py — UNREGISTERED

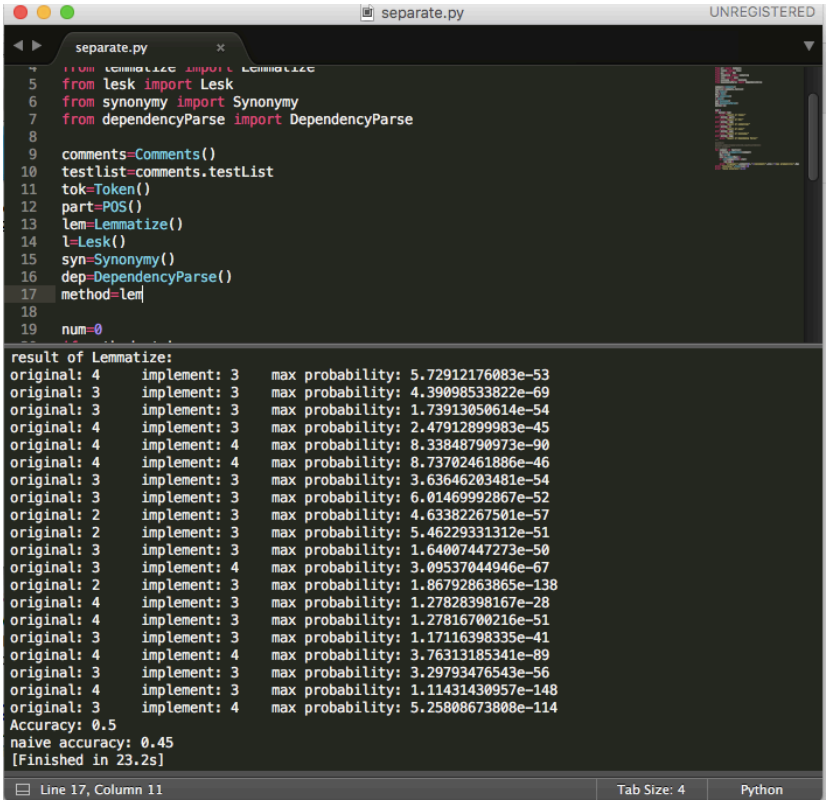
```
l=Lesk()
syn=Synonymy()
dep=DependencyParse()
method=tok

num=0
if method==tok:
    print "result of Token:"
elif method==part:
    print "result of POS:"
elif method==lem:
    print "result of Lemmatize:"
elif method==l:
    print "result of Lesk:"
elif method==syn:
    print "result of Synonymy:"
```

```
result of Token:
original: 4     implement: 3    max probability: 1.27092373556e-52
original: 3     implement: 3    max probability: 1.70587560863e-68
original: 3     implement: 3    max probability: 2.17033145223e-54
original: 4     implement: 3    max probability: 4.63206862393e-45
original: 4     implement: 4    max probability: 1.62433689358e-89
original: 4     implement: 4    max probability: 1.07524816229e-44
original: 3     implement: 3    max probability: 8.53211076103e-54
original: 3     implement: 3    max probability: 2.45265080103e-51
original: 2     implement: 3    max probability: 2.32897085096e-56
original: 2     implement: 3    max probability: 4.81810268584e-49
original: 3     implement: 3    max probability: 2.02789894184e-50
original: 3     implement: 4    max probability: 1.67824475361e-66
original: 2     implement: 3    max probability: 1.1291772361e-137
original: 4     implement: 3    max probability: 2.10404894581e-28
original: 4     implement: 3    max probability: 1.40990081372e-49
original: 3     implement: 3    max probability: 3.25331282506e-41
original: 4     implement: 4    max probability: 3.07183092852e-86
original: 3     implement: 3    max probability: 1.07730176519e-54
original: 4     implement: 3    max probability: 1.10360385018e-145
original: 3     implement: 4    max probability: 1.15572136211e-112
Accuracy: 0.5
naive accuracy: 0.45
[Finished in 20.0s]
```

Line 17, Column 11 — Tab Size: 4 — Python

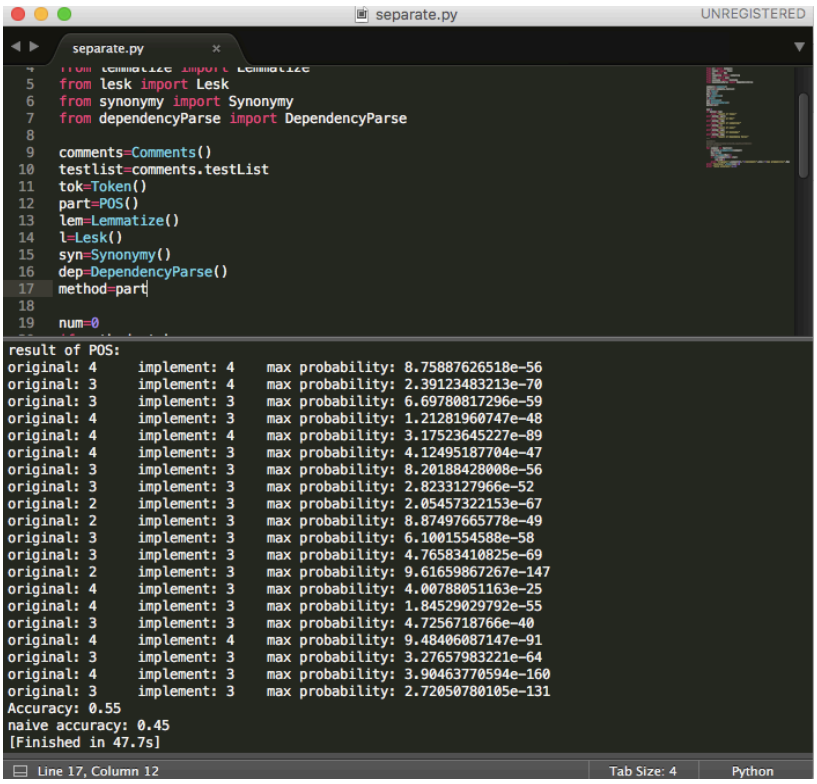
The accuracy of word token is 0.05 higher than naïve accuracy.

4.3 Lemmatization (Lexical Features)



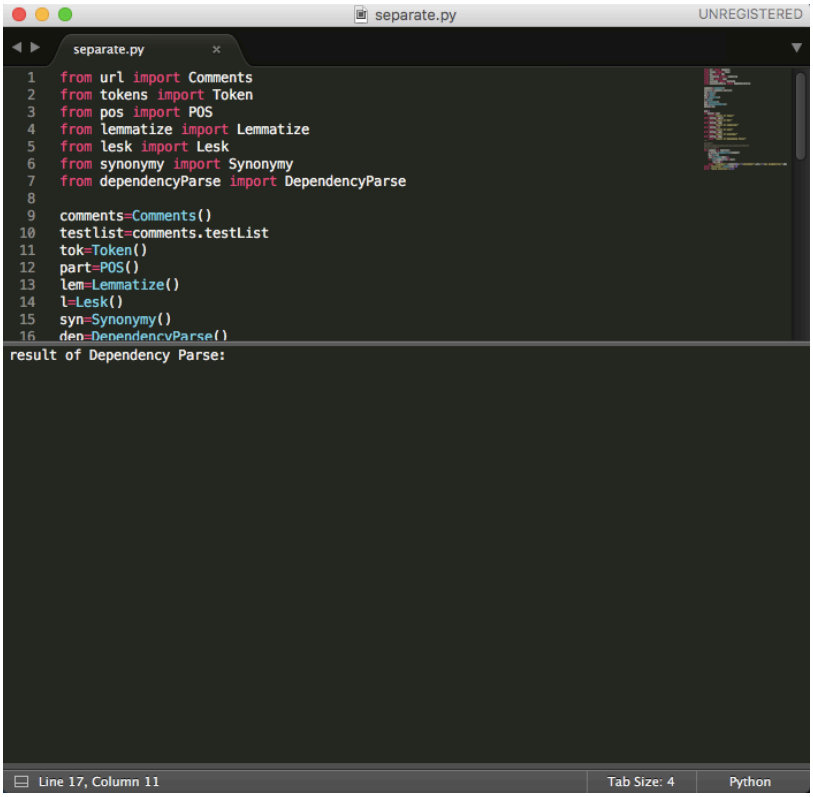
The accuracy of lemmatization is 0.05 higher than naïve accuracy.

4.4 POS Tagging (Syntactic Features)

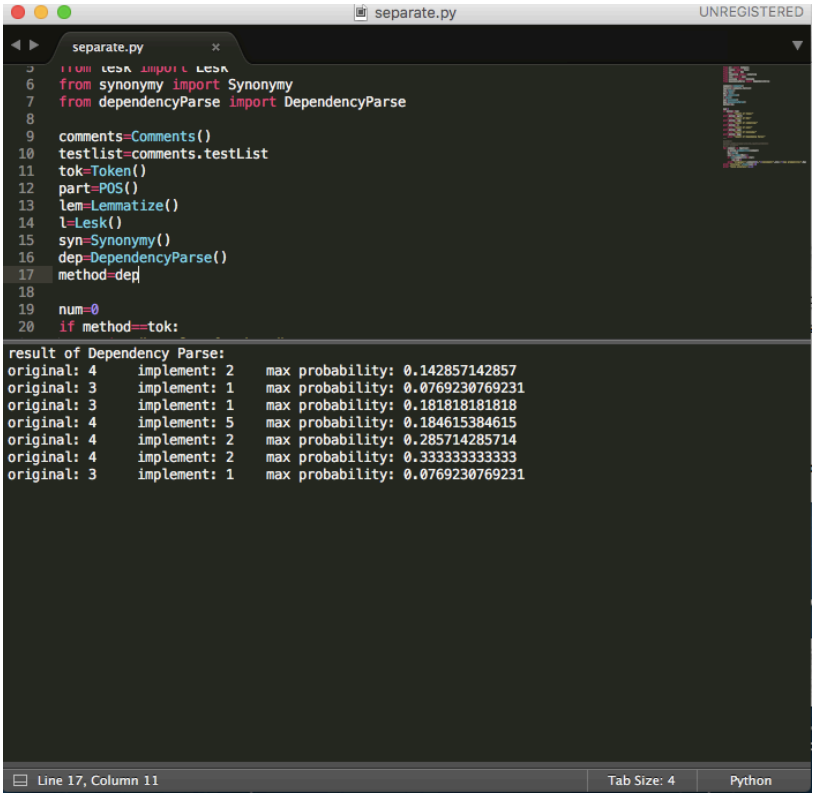


The accuracy of POS tag is 0.1 higher than naïve accuracy.

4.5 Dependency Parsing (Syntactic Features)



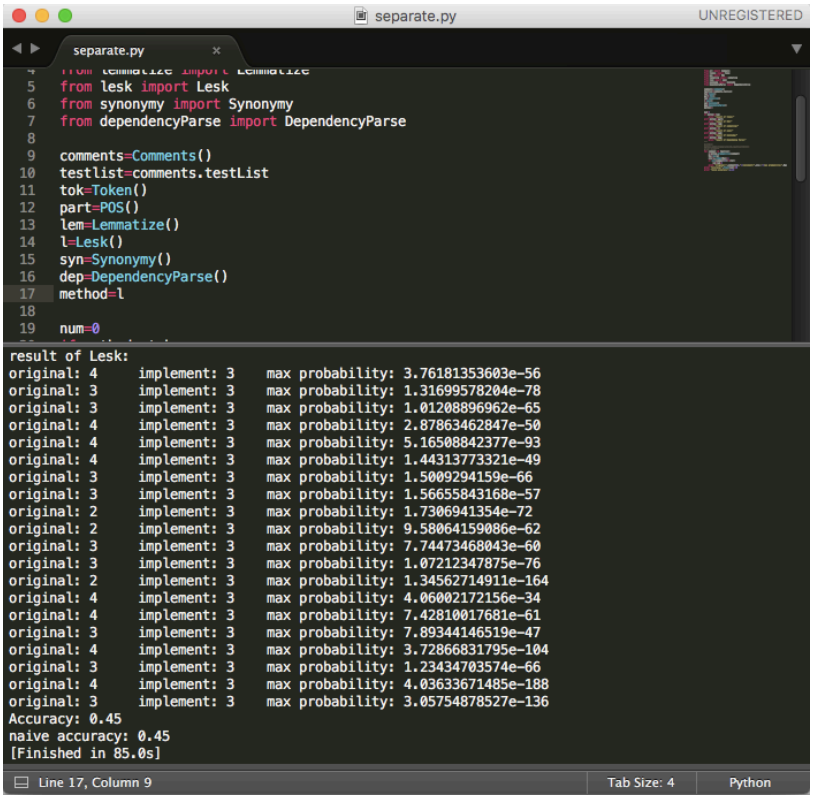
After running Dependency Parsing for a period of time, we get the result as picture above.



After running a whole night, we get part of the answers that we want.

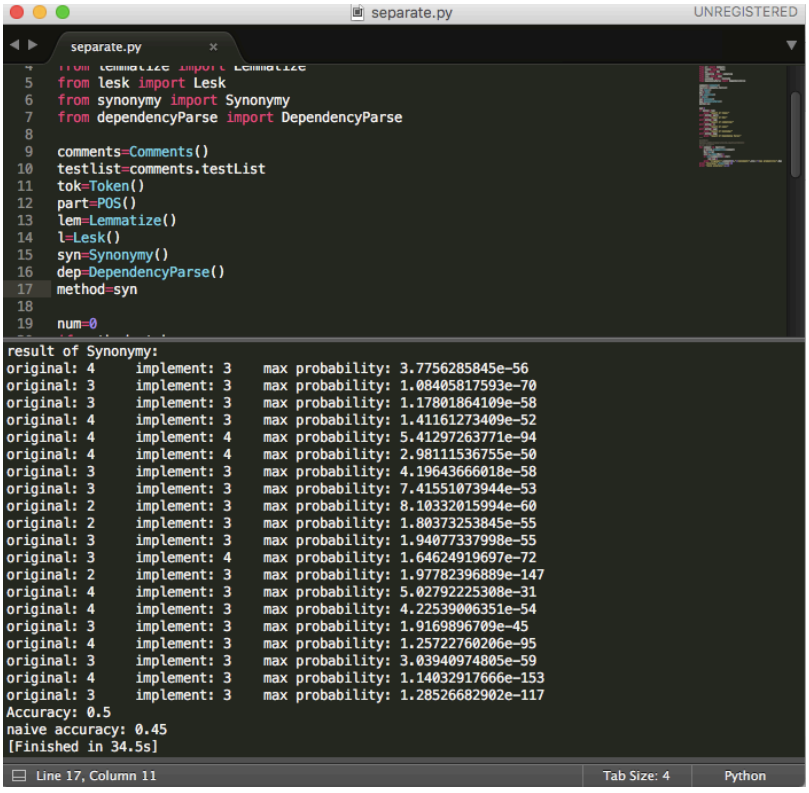
Therefore, we can finish dependency parsing but the running time is too large.

4.6 LESK (Semantic Features)



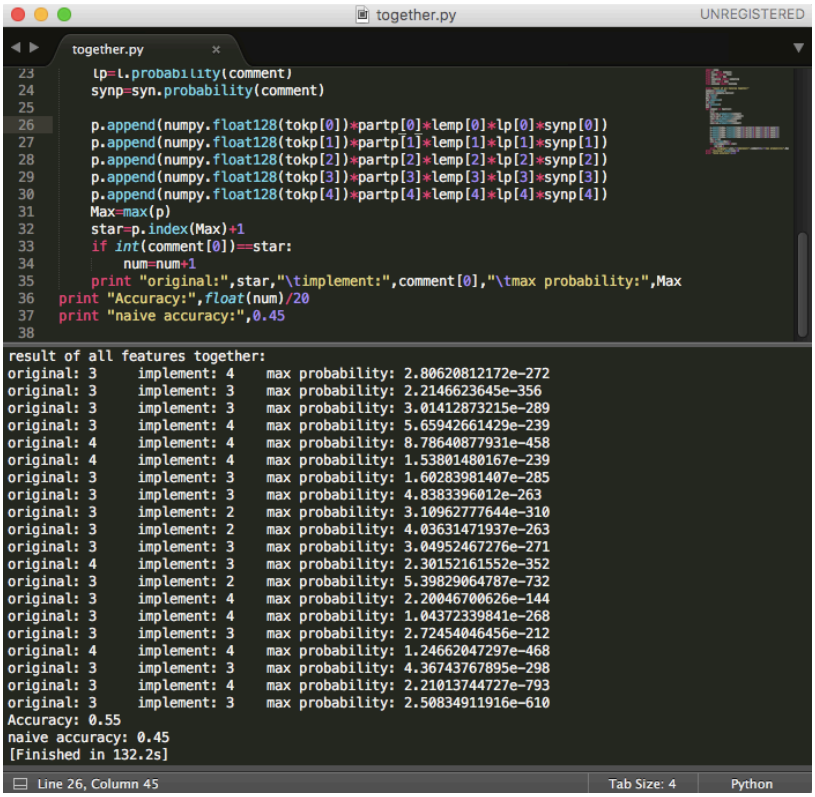
The accuracy of LESK algorithm is the same as naïve accuracy.

4.7 Semantic Relations (Semantic Features)



The accuracy of synonymy is 0.05 higher than naïve accuracy.

4.8 Six Features Together



The accuracy of using all feature is 0.05 higher than naïve accuracy.

5. Problems Encountered

We meet several problem during the whole project. First is Unicode. We cannot use Unicode directly. Therefore, we encoded Unicode to utf-8. Second problem is out of memory. When we parsing dependency of reviews in testing set, a warning came out, which is out of memory. Thus, we split sentences as single sentence and try again, which avoid this problem successfully. Third, once we run the program several times, we cannot visit yelp.com because of robotic policy. At this time, we change another IP address to solve this problem. At last, the maximum probability may display 0 because of overflow. So we update 64-bit to 128-bit.

6. Pending issues

In this six features, some of them take a long period of time. Dependency Parser, which belongs to syntactic feature, spend the longest running time. Therefore, we cannot run this feature in limited time. However, we still finish coding this feature and maybe we could use a better parser to make it faster.

7. Potential improvements

In this program, we need to implement the running time of each feature. For an Automatic Rating System, it takes too much time to analysis a review.