Natural Language Systems

Coursework 2

Alexander Angelov

Task 1: Named-entity recognition

a) Inaugular address corpus processed with NLTK NER

Firstly, all files of the inaugular address corpora are tokenized and tagged with their corresponding pos tag. Secondly, the nltk.ne_chunk function is performed to transform the corpora into a tree. The function is performed twice. The first time not allowing labels (presenting all recognized phrases with NE) and the second time allowing labels such as LOCATION, GPE and ORGANIZATION for recognizing specific phrase types.

b) Inaugular address corpus processed with Stanford named-entity recognizer

The inaugular tokenized corpus is inputted into the Stanford NER tagger (StanfordNERTagger). The operation is performed to the tokenized words of each file in the corpora. The tagger again recognizes specific phrases such as ORGANIZATION and LOCATION. The tagger also uses the tag "O" to state that the token is not part of any of the named entities.

c) Comparing the outputs of the two NER methods for the ORGANIZATION class

The tree results from a) are converted to a list via tree2conlltags function. The resulting list stores information that specifies the position of all words that are part of an ORGANIZATION phrase. They are 3 types: B-ORGANIZATION meaning that the token is the first word of the ORGANIZATION phrase, I-ORGANIZATION meaning that the token is part of the ORGANIZATION phrase and O meaning that the token is not part of such phrase. Then a list of only ORGANIZATION phrases (883 phrases in total) is created storing the whole phrase name and its exact location (ex. the triple ("Invisible Hand", 515, 516) indicates that the phrase is "Invisible Hand" and its tokens are the 515 and 516 words of the file). A similar list of the phrase name and its location (299 in total) is created when using the Stanford NER was applied. The results are then compared based on the phrase location. A total of 263 phrases were matched (exactly the same phrase on the exact location) and 112 have partial overlapping (if at least one word is part of the NLTK and the Stanford detected phrases), while 263642 did not match at all. An interesting observation is that almost all of the phrases that are detected from the Stanford NER have been matched.

Overall, the NLTK NER recognized more ORGANIZATION phrases when compared to the Stanford NER. However, most of the phrases detected by the NLTK NER do not detect the whole phrase. An example is the phrase "House of Representatives", the NLTK NER detects only the first word "House" of the ORGANIZATION entity. The NLTK NER is not capable of fully detecting phrases containing connection words such as "of" and "and". Instead, it detects only tokens that start with a capital letter. The Stanford NER on the other side detected many sophisticated phrases such as the "General Government of the Union", the "Bureau of Corporations", "Fountain of Justice" and the "Permanent Court of International Justice". It enables detecting phrases containing connection words, as well as entities that start with a small letter such as "interior administration". This results in detecting more realistic phrases. Overall, the quality of the Stanford NER is much better than the NLTK one.

Task 2: Sentiment analysis of movie reviews

a) Bootstrapping sentiment lexicon

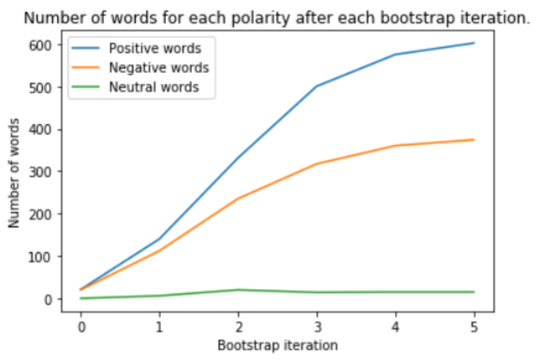
In the beginning, infinity and minus infinity values are given to the adjectives that are known to be positive (infinity) or negative (minus infinity). Then 3 major patterns were created for increasing the lexicon, each accepting only JJ values to be added to the lexicon. All of the patterns take into consideration whether adverbs are placed before the adjective. This enables us to detect more adjectives and to cover more patterns such as "but also" and "but not".

- **"JJ_1 (if not (RB)^n JJ_2"** ; **"JJ_1 , if not (RB)^n JJ_2"** ; **"JJ_1 : if not (RB)^n JJ_2"** ; **"JJ_1 if not (RB)^n JJ_2"**
  The 4 "if not" patterns were implemented to detect adjectives from the opposite polarity. Both JJ1 and JJ2 can be detected even when adverbs are placed between them. The pattern classifies many adjectives correctly such as "goofy (if not entirely wholesome". However, due to its dual meaning, it produces a small number of errors. An example is the phrase "good, if not entirely fresh" which contains two adjectives from the same polarity.
- **"JJ_1 (but (RB)^n JJ_2"** ; **"JJ_1 , but (RB)^n JJ_2"** ; **"JJ_1 : but (RB)^n JJ_2"** ; **"JJ_1 but (RB)^n JJ_2"**
  The 4 "but" patterns were implemented to detect adjectives from the opposite polarity. Both JJ1 and JJ2 can be detected even when adverbs are placed between them. The pattern classifies many adjectives correctly due to the nature of the word "but" (ex. "ditsy but heartfelt"). However, it produces a small number of inconsistencies. For example, the phrase 'predictable but still entertaining' is contrasting a neutral word with a positive one.
- **"JJ_1 , (RB)^n JJ_2 , (RB)^m JJ_3, …. , (RB)^l JJ_p and (RB)^m JJ_p+1"** ; **"JJ_1 , (RB)^n JJ_2 , (RB)^m JJ_3, …. , (RB)^l JJ_p , and (RB)^m JJ_p+1"**
  A pattern for finding adjectives with the same polarity. The pattern detects adjectives part of a sequence even when adverbs are placed between them. This pattern discovers many new adjectives. For example, "simplistic, silly and tedious" phrase can detect two adjectives from a single one. However, it produces many errors due to its ability to skip adverbs. For example, in the phrase "overlong and not well-acted" two words of different polarities are classified as the same polarity, because the adverb not was skipped.

In addition, an error that occurs in all patterns is that a neutral word may be classified as a positive or negative.

For all patterns, when a new word is discovered, it is put into the lexicon with a value of -1(if negative) or 1(if positive). If the word already exists, its value is increased with 1(if positive) or decreased with -1(if negative).

A bootstrapping is performed 5 times to increasing the positive and negative adjectives (with each of the implemented patterns). In each of its iterations, the patterns are applied to all adjectives in the lexicon to discover more adjectives. This has increased the lexicon size drastically. This change can be seen in the following figure.

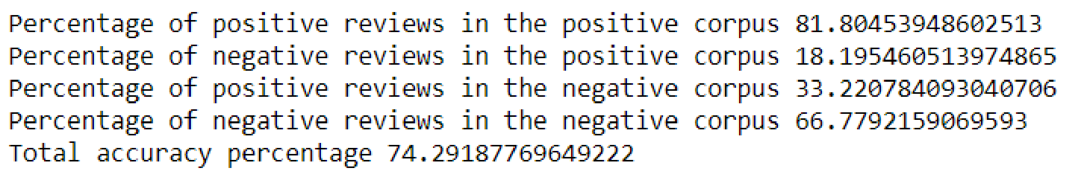


From the figure, it can be observed that bootstrapping is producing more positive adjectives than negative ones (almost 2 times more).

After the last bootstrap iteration, the lexicon was compared to the MPQA lexicon. It produces a 35% correctness which is very low even though the patterns recreate patterns from the English grammar. The low percentage is received maybe because an adjective is wrongly classified in the first iterations. This may have created the opposite effect and created the low correctness percentage. Another reason for the low percentage is that many of the adjectives do not appear in the MPQA lexicon at all.

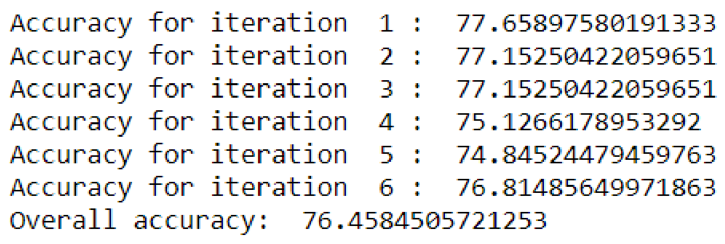
b) Classification and evaluation

- Baseline
  Firstly, an MPQA lexicon is created from the given MPQA. Then, for each review, the positive and negative words are calculated and compared. The review is classified as positive if it contains more positive words and vice versa to negative if it is containing more negatives. If their number is equal, it is chosen at random. The overall accuracy is 74.29%. A more detailed set of results is presented in the following figure.



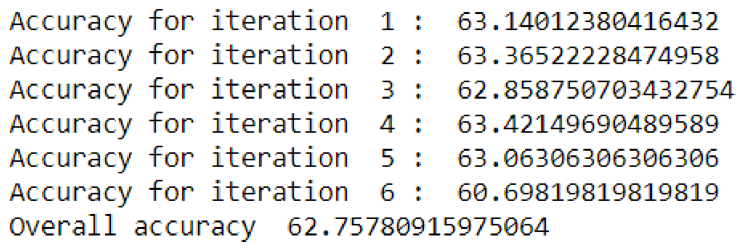
```
Percentage of positive reviews in the positive corpus 81.80453948602513
Percentage of negative reviews in the positive corpus 18.195460513974865
Percentage of positive reviews in the negative corpus 33.220784093040706
Percentage of negative reviews in the negative corpus 66.7792159069593
Total accuracy percentage 74.29187769649222
```

- Bow classifier
  A BoW matrix was created from all reviews with the CountVectorizer from scikit. Then a 6-fold validation is performed with logistic regression for the classification task. The average accuracy of all iterations is 76.46%. When compared to the baseline it can be observed that the BoW method is producing more accurate results (with around 2% better performance). The accuracies of all iterations can be seen in the following figure.



```
Accuracy for iteration  1 :  77.65897580191333
Accuracy for iteration  2 :  77.15250422059651
Accuracy for iteration  3 :  77.15250422059651
Accuracy for iteration  4 :  75.1266178953292
Accuracy for iteration  5 :  74.84524479459763
Accuracy for iteration  6 :  76.81485649971863
Overall accuracy:  76.4584505721253
```

- Word2vec classifier
  The word2vec is used from gensin for all reviews to create word2vec embeddings. For each review, the embeddings of the words in the review are averaged. Then a 6 fold-validation is performed with logistic regression for the classification task. The average accuracy of the approach is 62.76%. This accuracy is much lower compared to the accuracies of the baseline and the BoW approach. A more detailed list of results is provided in the following figure.



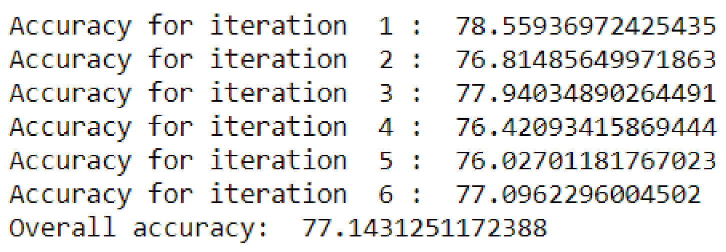
```
Accuracy for iteration  1 :  63.14012380416432
Accuracy for iteration  2 :  63.36522228474958
Accuracy for iteration  3 :  62.858750703432754
Accuracy for iteration  4 :  63.42149690489589
Accuracy for iteration  5 :  63.06306306306306
Accuracy for iteration  6 :  60.69819819819819
Overall accuracy  62.75780915975064
```

- Additional features

  Four additional feature were applied to both approaches
  - Number of negations (How many of the words no, not and never appear in the review)
  - The number of positive words from the MPQA lexicon
  - The number of negative words from the MPQA lexicon
  - Removing numbers from the corpus

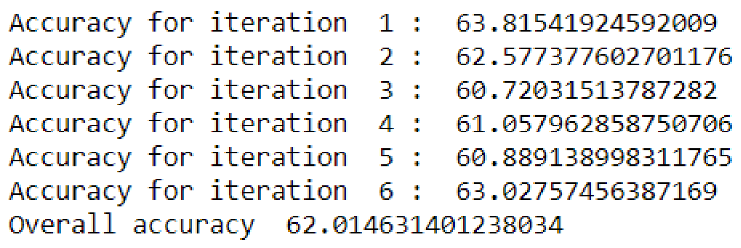
  After applying the additional features, the following results are observed:

  Average of 77.14% when the BoW approach is used. This is a slight improvement of 2% when compared to the average accuracy when no additional features are included.



```
Accuracy for iteration  1 :  78.55936972425435
Accuracy for iteration  2 :  76.81485649971863
Accuracy for iteration  3 :  77.94034890264491
Accuracy for iteration  4 :  76.42093415869444
Accuracy for iteration  5 :  76.02701181767023
Accuracy for iteration  6 :  77.0962296004502
Overall accuracy:  77.1431251172388
```

  Average of 62.01% when the word2vec approach is used. It produces almost the same set of results as when no additional features were included.



```
Accuracy for iteration  1 :  63.81541924592009
Accuracy for iteration  2 :  62.577377602701176
Accuracy for iteration  3 :  60.72031513787282
Accuracy for iteration  4 :  61.057962858750706
Accuracy for iteration  5 :  60.889138998311765
Accuracy for iteration  6 :  63.02757456387169
Overall accuracy  62.014631401238034
```

- Discussion and ideas

  Overall, the BoW approach produces the most accurate results when additional features were added. It is very efficient because it detects words that are often surrounding the token. However, its efficiency depends on the range of the context (context window) that was used. Also, it is memory-consuming and very slow especially when a large vocabulary has been provided.
  Word2vec does not perform very well on the provided data. However, it is much faster than the BoW approach and much more memory-friendly. One of the biggest drawbacks is that it does not recognize unfamiliar tokens. Another issue is that all of the meaning of the words are presented by 1 vector (ex. second can be a time of 1 second or finishing on second place).
  In addition, several approaches can be observed in the future. Firstly, other classifiers such as the Bayes classifier can be applied for the classification. Secondly, the BoW and word2vec approaches can be combined in order to produce more accurate results. Thirdly, bag-of-n-grams and GloVe methods can be used for the classification because take less time for training. Also, more additional features can be added to improve the accuracy of the model. Examples of such features include the consideration of words pos tags and the identification of words that are affected by negations.