Coursework 2 - NLS

9911494 - Anca Radulian

April 3, 2020

1 Introduction

This report presents the functionality implemented in the coursework, the results obtained, and their analysis. In the coursework, the nltk python library was used to import, parse, and tokenize the documents, as well as to perform NER. The StanfordCoreNLP collection was also used to perform NER. The LogisticRegression, and accuracy_score, KFold and CountVectorizer are part of the `sklearn` library and were used to construct and validate a sentiment classifier.

2 Part 1c: Named Entity Recognition

In the previous sub-tasks of part 1, two different tools were used on a given corpus to find and classify the entities from documents. The first one is from the `nltk` library called `ne_chunk()` which returns a nested `nltk.Tree` object. To parse this object and collect only the ORGANIZATION entities, the method `tree2conlltags()` was used which converts a `nltk.Tree` to a list of 3-tuples containing `(word, pos, IOB-tag)`. Since the lists contains IOB-tags, another parsing is necessary to combine in a single chunk the B- and I- tags only for ORGANIZATION entity. The second tool used is Stanford CoreNLP collection from which the NER annotator was used. The `StanfordCoreNLP.ner()`method returns the annotation result in a list of tuples containing `(word, tag)`. Even though the output is in a format easier to parse, it lacks information as the models and rules files that are part of the Stanford NER tool use a basic IO tagging scheme, compared to the IOB-tagging scheme from the `nltk` tool. Moreover, to use the Stanford tool an extra step has to be taken to start a server and also, the execution time is significantly longer compared to the `nltk.ne_chunk()` annotation.

Observations made based on the output:

- The `nltk` tool classified 877 tokens as ORGANIZATION, while the Stanford tool only 398. The Stanford tool missed some of the entities and the nltk added many entities tagged wrongly. E.g: nltk tool wrongly classified 'Fellow Citizens', 'Divine Author', 'Atlantic', 'Great Lakes', etc. as ORGANIZATION entities, whereas the Stanford tool did not. The Stanford tool missed a few entities which the nltk identified (e.g. no

ORGANIZATION entity was found in 1805-Jefferson, 1865-Lincoln and 1973-Nixon files)

- Fully matches: 216
- Partial matches: 75

A *fully match* was considered when the the word and the index of that word in the file were matching in both result lists (e.g. ('Senate', 3) and ('Senate', 3)). A partial match was considered if a segment of an ORGANISATION chunk and its index was matched with an entity form the other list (e.g. the nltk tool retrieved 'Federal Government' but the Stanford tool only found 'Federal' or 'Government'). However, these partial matches mostly happened because the Stanford `ner()` method does not use the IOB tagging scheme by default. Therefore it is unable to provide boundaries to entities compared to the NLTK.

As a conclusion, the Stanford tool seems to be better at classifying and not adding false positives, but is more difficult to use in contrast to the nltk tool, and it takes considerably longer to produce the output.

3 Part 2a: Sentiment/ Polarity Lexicon

In this part of the coursework, a lexicon of positive and negative adjectives was extended by following certain patterns. The pattern implemented in this task and their issues are:

- *(RB) JJ, (RB) JJ, ...and (RB) JJ* - enumeration of adjectives where adverbs are allowed in between adjectives. All adjectives in the enumeration are considered to have the same polarity (e.g. from corpus 'simplistic , silly and tedious' or 'grand , uncomplicated fashion'). This pattern is good at finding many adjectives, but the main issue is the assumption that all adjectives have the same polarity which is not entirely correct. This assumption classifies a lot of adjectives wrongly. For instance in the phrase 'funny , yet ultimately cowardly' - 'funny' is a positive adjective and 'yet' and 'ultimately' are both adverbs. Thus, the adjective 'cowardly' is classified as positive. Another error case is when an adjective with no polarity is added, e.g. 'direct-to-video' from 'direct-to-video stuff' was classified as negative, because 'stuff' was added in the negative lexicon due to a POS tagging issue. Also this pattern introduces lots of adjectives in both the positive and negative lexicon (1508 common adjectives).
- *JJ but (RB) JJ* - this pattern assumes that the two adjectives have opposite polarities. This pattern is adding many correctly classified adjectives, with only a few wrongly ones such as 'enough' which is neutral adjective from the phrase 'good but enough'.
- *JJ but not (RB) JJ* - this pattern assumes that the two adjectives have same polarities since we have negation in front of the second adjective. Same as the above pattern, it produces mostly correct classifications. One wrong example would be the word 'frenetic' classified as positive from the phrase 'frenetic but not really funny'.

To solve the problem with the conflicting polarities of adjective, the frequency of the words in both lexicon was used. However, probably due to the enumeration pattern, there was a big number of adjectives with identical frequency (1218) and in the equality case all the adjectives were assigned the positive polarity. The final count of adjective was: 1362 positives and 403 negatives. A proper count of how many adjectives were classified correctly is hard to compute as the lexicon is quite big, but I would mention that because the conflicting polarity issue was left at the end, many adjectives were erroneously classified.

4 Part 2b: Sentiment Classifier

For the baseline approach the classification of each review was done by assigning the polarity based on whether there were more positive or more negative words from the MPQA lexicon. An improvement here would be to take into consideration the POS tag of the word as well when counting the words. This could be done in consideration of the MPQA lexicon having multiple entries for the same word for different tags (the word 'will' when is tagged as verb has no polarity, compared to when it is tagged as a noun or adjective). The percentages of accuracy of this approach were the following:

```
Accuracy for positive reviews: 81.80%
Accuracy for negative reviews: 41.44%
Overall accuracy: 61.62%
```

The machine-learning approach was implemented using the following features: bag of words, the count of positive and negative words from MPQA lexicon and the number of negations (no, not, never and n't) present in a review. The BOW feature is done using `CountVectorizer()` which returns a sparse matrix. The two other features are computed using the review files and then appended to the BOW matrix. The `K-Fold(n_splits=5)` method is used to generate splits for the training and testing data. After researching a few different machine learning frameworks and analysing the level of accuracy, I have selected the `LogisticRegression()`for the sentiment classifier.

Accuracies using `KNeighborsClassifier` model

```
Accuracy of classifier: 65.54%
Accuracy of classifier: 67.18%
Accuracy of classifier: 65.06%
Accuracy of classifier: 65.90%
Accuracy of classifier: 63.79%
Overall accuracy of classifier: 65.49%
```

Accuracies using `RandomForestClassifier` model:

```
Accuracy of classifier: 72.90%
Accuracy of classifier: 75.48%
Accuracy of classifier: 73.08%
Accuracy of classifier: 74.58%
Accuracy of classifier: 73.12%
Overall accuracy of classifier: 73.83%
```

Accuracies using `LinearSVC` model:

```
Accuracy of classifier: 76.04%
Accuracy of classifier: 75.53%
Accuracy of classifier: 74.86%
Accuracy of classifier: 75.84%
Accuracy of classifier: 75.98%
Overall accuracy of classifier: 75.65%
```

Accuracies using `LogisticRegression`model:

```
Accuracy of classifier: 77.17%
Accuracy of classifier: 77.97%
Accuracy of classifier: 76.22%
Accuracy of classifier: 78.75%
Accuracy of classifier: 76.74%
Overall accuracy of classifier: 77.37%
```

Based on these findings, it becomes apparent that the machine-learning approach is more efficient than the baseline implementation based on the level of accuracy. Further improvements can be done to the negation feature by creating an analysis using the following heuristic: explore what words are affected by the negation and how this changes the polarity of that phrase.