Natural Language Processing - Assignment 2
Sentiment Analysis for Movie Reviews

Deadline: 24 September, 2020
Questions: nlp-course@utwente.nl
Sentiment analysis, i.e. determining whether a text is positive or negative, is a challenging task for computers. In this assignment, we will see some simple techniques for doing so, using a database of movie reviews as our text. We will build statistical models and use them to predict the sentiment of a review, based on the words used.
Being able to automatically classify positive and negative texts has multiple practical applications. Besides being used for marketing (for example, companies use sentiment analysis on twitter feeds to determine how people react to their products), it has recently been used to automatically find people reporting bad reactions to drugs in their Facebook messages, and is a common component of many complex NLP-based applications.
The assignment is meant to be done by completing the Python 3 Notebook provided on Canvas. You may do so using Jupyter Notebook or Google Colab.

How to submit

Before you get started, here's how to submit your homework.

- Put your answers to the questions below in one PDF document.
- Clearly state your name(s) at the top of the document.
- Put the document and your Python Notebook (not the data files!) in one zip file.
- Use your group name in the name of the zip file.

1 Manual classification (0.5 pt)

To get started, read these two reviews, and decide which one is negative and which one is positive. Provide a short motivation, and try to anticipate what could pose an issue for automatic sentiment identification.

R1 Busy Phillips put in one hell of a performance, both comedic and dramatic. Erika Christensen was good but Busy stole the show. It was a nice touch after The Smokers, a movie starring Busy, which wasnt all that great. If Busy doesnt get a nomination of any kind for this film it would be a disaster. [...]
R2 This movie was awful. The ending was absolutely horrible. There was no plot to the movie whatsoever. The only thing that was decent about the movie was the acting done by Robert DuVall and James Earl Jones. Their performances were excellent! The only problem was that the movie did not do their acting performances any justice. [...]

What to submit: Your guesses, motivation & possible issues for automatic sentiment identification.

Dataset

In this assignment you will use a dataset of movie reviews from the Internet Movie Database (IMDB)[1].
The `movies` directory contains two subdirectories:

- `train` These documents will be used to train your language model. (600 docs)
- `test` These documents will be used to test your model. (50 docs)

The documents are named as `[sentiment]-[review ID].txt`. The text file `groundtruth.txt` contains the correct labels for the documents in `test`.
There is no need to modify the files or their directories. Load them using the provided Python 3 Notebook.

2 Tokenization

The first step is to tokenize the data. Tokenization splits up a character sequence into smaller pieces (tokens). An example tokenization is:
Original sentence: "If you have the chance, watch it. Although, a warning, you'll cry your eyes out."
Tokens: [`If`, `you`, `have`, `the`, `chance`, `,`, `watch`, `it`, `.`, `Although`, `,`, `a`, `warning`, `,`, `you`, `'ll`, `cry`, `your`, `eyes`, `out`, `.`]

2.1 Making your own tokenizer (0.5 pt)

For this assignment, make a simple tokenizer. Write 3 sentences and try the tokenizer out on them.

What to submit: Provide a description of how your tokenizer works. Report the tokens you obtain when using your tokenizer on your example sentences.

2.2 Using an off-the-shelf tokenizer (1 pt)

Compare the tokenizer you implemented in the previous question with one from NLTK, using the sentences provided in the Notebook.

What to submit: Reflect and answer these questions: What are the differences in the two tokenizer outputs? Which one is better? While coding your tokenizer, did you foresee all these inputs? Is there a single 'perfect tokenizer'?

2.3 Vocabulary (1 pt)

Run the NLTK tokenizer on all documents in the `train` directory and keep track of the unigram frequencies. Since our dataset is small, it is a good idea to apply heavy normalizations, for example removing punctuation and transforming each

[1]The dataset is a subset of the original dataset by Maas et al. The full dataset can be found at `http://ai.stanford.edu/~amaas/data/sentiment/`.

sentence to lowercase. After you implement the normalization, the sentence "If you have the chance, watch it. Although, a warning, you'll cry your eyes out." should look similar to this:
Normalized tokens: [`if`, `you`, `have`, `the`, `chance`, `watch`, `it`, `although`, `a`, `warning`, `you`, `'ll`, `cry`, `your`, `eyes`, `out`]

Answer the following questions using the documents in the `train` directory:

- How many unique n-grams are there? (where n=1,2,3).
- Report the top 10 most frequent words (unigrams) and their frequencies. What kind of word are these?
- How many words occur 1, 2, 3, 4 times in the corpus? Which kind of distribution is this?

Since words that do not occur often don't add much information to the classification, keep only the words that occur at least 25 times as your vocabulary. Write your code such that all words that are not in your vocabulary are ignored in the rest of this assignment.

What to submit: Answer to the above 3 points.

3 Text classification with a unigram language model (2,5 pt)

Recall that for a text with words $w_1 \dots w_n$, we calculate the probability as follows using a unigram language model:

$$P(w_1, w_2, ..., w_n) = P(w_1)P(w_2) \dots P(w_n) = \prod_{i=1}^{n} P(w_i)$$

In order to avoid underflow, it is better to calculate this in log space (base 2)[2]:

$$\log P(w_1, w_2, \dots, w_n) = \log \prod_{i=1}^{n} P(w_i) = \sum_{i=1}^{n} \log P(w_i)$$

In our dataset we have two classes: positive (Pos) and negative (Neg). For each class, we will calculate a separate language model. This is the training or learning phase. In the apply phase, we will classify new texts as positive or negative. For testing our machine learning classifier, we apply the models on the documents in the test part of the corpus.

1. **TRAIN** For the documents in the `train` directory, build two language models. One using the positive reviews, and one using the negative reviews. For example, we calculate the probability for the positive language model as follows.

$$P(w_1, w_2, ..., w_n|Pos) = \prod_{i=1}^{n} P(w_i|Pos)$$

[2]Since the probabilities are "small numbers", the more we multiply them together, the smaller they become, up to a point where the computer cannot represent these number accurately anymore (`https://en.wikipedia.org/wiki/Arithmetic_underflow`). By moving everything in log space we sidestep the problem (recall that the logarithm of a product is the *sum* of the individual logarithms).

Where we are using the conditional probability ($P(w_i|Pos)$ instead of just $P(w_i)$), because we are calculating the probabilities using only positive reviews. We estimate the conditional probabilities:

$$P(w_i|Pos) \approx \frac{C(w_i, Pos)}{\sum_{w \in V} C(w, Pos)}$$

where $C(w_i, Pos)$ is the frequency of word w_i in the positive reviews and $\sum_{w \in V} C(w, Pos)$ the total number of words in the positive reviews[3].

Smoothing Use smoothing to avoid zero probabilities:

$$P(w_i|Pos) \approx \frac{C(w_i, Pos) + k}{\sum_{w \in V} C(w, Pos) + kV}$$

Where V is the size of your vocabulary. Use two settings: when $k = 1$ and a value for k that you have selected yourself.

2. **TEST** For the reviews in the `test` directory, calculate the probability for both language models. Assign each review the class for which it has the highest probability. Using the MAP (Maximum Aposteriori Probability) rule:

$$Class(R) = \arg\max_{C} P(C|R)$$

Evaluation The last cell of the Notebook will compute the accuracy of your predictions on the documents in the test set. As an indication, your accuracy should be above 60%. If it is lower than that (or higher than 80%) there is a big chance that you have a bug in your code. [4]

What to submit: The performance of your classifier (accuracy) for both runs (smoothing with $k = 1$, and a k that you have selected yourself).

Optional background information: The method presented here is in fact the same as a multinomial naive Bayes classifier, with equal prior class probabilities. In our case, this makes sense, since we expect the proportion of positive and negative reviews to be equal. In case this is not realistic we can estimate the prior class probabilities $P(Class = Pos)$ and $P(Class = Neg)$ from the corpus, using maximum likelihood estimation.

4 Text classification with a bigram language model (3 pt)

Now do the same as in section 3 over again, but create a classifier that computes the probability of a review $w_1, w_2, ..., w_n$ as:

$$\log P(w_1, w_2, \ldots, w_n) = \log P(w_1| < S >) + \sum_{i=2}^{n} \log P(w_i|w_{i-1})$$

[3] Take a few minutes to understand the formula, and reflect on the meaning of this sentence.

[4] A higher performance does not automatically result in a higher grade. This assignment is about understanding the concepts and understanding how to apply them, rather than the mere performance of your classifier.

Take care not to enforce the threshold of 25 occurrences also for the bigrams, as this is too high for our small dataset.
Notice that, when we are dealing with bigrams, we want to know the probability of a word knowing that we have seen the previous one (*conditional probability*), and this is different from knowing the probability of the two words at the same time (*joint probability*).
Imagine that in our dataset the word "amazing" is almost always followed by "actress". In this case, "amazing actress" (*joint probability*) will be just one of the many bigrams we have, hence $P(\text{"amazing actress"}) = 0,00000....$ On the other hand, if we know that the current word is "amazing" (*conditional probability*), the probability that the next work is "actress" is very high: $P(\text{"actress"}|\text{"amazing"}) = 0.999....$ Take a minute to reflect on what this means for your classifier, and for the formulas we saw earlier.

What to submit:

- Report how you compute $p(w|w')$
- What is the accuracy of this second model on the test?

5 Improving the classifier (1,5 pt)

Discuss your results and ways that could improve the classifier. Are there other types of features that could improve the classification? What are the disadvantages of the current way of normalizing and preprocessing the text? Suggest possible changes. Motivate your suggestions, for example based on observations during the error analysis.
Additional bonus points can be awarded if you implement one suggestion (or more), test if it improves the performance, and discuss why (or why not).

What to submit: Few (2-4) suggestions for improvements and your motivations. (Optional, for maximum points: a new unigram classifier with your implemented suggestion, its performance, and comments on its result).