Natural Language Processing 2025-2026

Homework 2: Sentiment Analysis for Movie Reviews

Daisy Baars (s2556510)
Kornel Palkovics (s3698920)
Group 59
17/09/2025

1 Tokenization

1.1 Making your own tokenizer (0.5 pt)

For this assignment, we made a simple tokenizer and tested it for three sample sentences. First, the text will be transformed into all lowercase with text.lower. After this, the punctuation is removed to only end up with the separate words, which will be split on the white spaces.

We tested the tokenizer with the next four example sentences:

sample_string0 = "If you have the chance, watch it. Although, a warning, you'll cry your eyes out."

sample_string1 = "kaas is lekker"
sample_string2 = "Me and My bEstfriend are going to the city!"
sample_string3 = "me and my number 3 bEStfrienD like movies"

This gave us the following tokens as outcome:
['if', 'you', 'have', 'the', 'chance', 'watch', 'it', 'although', 'a', 'warning', "you'll", 'cry', 'your', 'eyes', 'out']
['kaas', 'is', 'lekker']
['me', 'and', 'my', 'bestfriend', 'are', 'going', 'to', 'the', 'city']
['me', 'and', 'my', 'number', '3', 'bestfriend', 'like', 'movies']

1.2 Using an off-the-shelf tokenizer (0.5 pt)

There are multiple differences between the two tokenizer methods. First of all, the NLTK tokenizer does not lowercase all words like my_tokenizer does. Lowercasing reduces the amount of tokens (because *"Friend"* and *"friend"* will be the same), however it does lose context information. The NLTK tokenizer also keeps the punctations and emojis. The my_tokenizer keeps contractions like *"won't"* while the NLTK tokenizer separates this in *"wo"* and *"n't"*

We did not foresee all these inputs while coding. We did not purposely delete emojis and emoticons. However, we did choose to put everything

in lowercase and not tokenize the punctuations to keep the vocabulary concise.

The "perfect" tokenizer mostly depends on the task and context you want to use the tokenizer for. For example, if you need a lot of contextual information, it may be better to choose to keep all punctuation and emojis, but if you want to keep your vocabulary concise, you can choose to only tokenize the words. This means there is not one perfect tokenizer that suits all scenarios and needs.

2 Text classification with a unigram language model

2.1 Theory (2 pt)

Exercise 2.1.1

First, we have the equation for P(w_i | Pos) with Laplace smoothing:

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

Then, apply this to the probability of P(*boring* | *Pos*)

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

The first step is to fill in all the given variables.

$k = 1$

$V = 49$ (the whole corpus, negative and positive)

$C(boring, Pos) = 0$ (*boring* does not appear in the positive reviews)

$\sum_{w \in V} C(w, Pos) = 43$ (5+3+10+25, because *the* = 5 times, *a* = 3 times, *really, plot, movie, great, are* = each 2 times (10) and 25 other tokens that appear once in the positive corpus)

$$P(boring|Pos) \approx \frac{0 + 1}{43 + 49} \approx \frac{1}{92} \approx 0.0108696$$

Exercise 2.1.2

To decide if "intriguing yet disappointing" will be classified as a positive or negative review, we first have to compute the probability for each word in each class. After that, we use the bag-of-word model and, to simplify, assume the words are mutually independent. First, we will compute the probability that "intriguing yet disappointing" is a positive review, and then we will compute the probability that the sentence is a negative review. Lastly, we will compare the negative and positive review probability for the sentence and decide how it would be classified.

Calculation of $P(intriguing\ yet\ disappointing \mid Pos)$:

$P(intriguing|Pos) \approx \frac{1+1}{43+49} \approx \frac{2}{92}$ (*intriguing* is in the list of tokens that appears once in the positive reviews, which is why it $C(intriguing, Pos)$ gets a score of 1.
$P(yet|Pos) \approx \frac{0+1}{43+49} \approx \frac{1}{92}$ (*yet* is not in the list of tokens that appear in the positive reviews, which is why it $C(yet, Pos)$ gets a score of 0.
$P(boring|Pos) \approx \frac{0+1}{43+49} \approx \frac{1}{92}$ (*boring* is not in the list of tokens that appear in the positive reviews, which is why it $C(boring, Pos)$ gets a score of 0.

$$P(intriguing|Pos) * P(yet|Pos) * P(boring|Pos) \approx \frac{2}{92} * \frac{1}{92} * \frac{1}{92} \approx \frac{2}{92^3}$$
$$\approx 0.000002568$$

Concluding, this means $P(intriguing\ yet\ disappointing \mid Pos) \approx 0.000002568$

Calculation of $P(intriguing\ yet\ disappointing \mid Neg)$:
$P(intriguing|Neg) \approx \frac{0+1}{34+49} \approx \frac{1}{83}$ (*intriguing* is in the list of tokens that appear once in the negative reviews, which is why it $C(intriguing, Pos)$ gets a score of 1.
$P(yet|Neg) \approx \frac{0+1}{34+49} \approx \frac{1}{83}$ (*yet* is not in the list of tokens that appear in the negative reviews, which is why it $C(yet, Pos)$ gets a score of 0.
$P(boring|Neg) \approx \frac{1+1}{34+49} \approx \frac{2}{83}$ (*boring* is not in the list of tokens that appear in the negative reviews, which is why it $C(boring, Pos)$ gets a score of 0.

$$P(intriguing|Neg) * P(yet|Neg) * P(boring|Neg) \approx \frac{1}{83} * \frac{1}{83} * \frac{2}{83} \approx \frac{2}{83^3}$$
$$\approx 0.000003497$$

Concluding, this means $P(intriguing\ yet\ disappointing \mid Neg) \approx 0.000003497$

Final conclusion:
Since $P(intriguing\ yet\ disappointing \mid Neg) > P(intriguing\ yet\ disappointing \mid Pos)$, the sentence will be classified as a negative review.

2.2 Coding (2 pt)

The accuracy of the classifier when running with the Laplace smoothing is 0.84, whilst without Laplace smoothing the accuracy is 0.5 (see Notebook for code).

The accuracy of the classifier with Laplace smoothing and removing the stop words is 0.85, which is 0.01 higher than without removing them. Stop words appear in both classes almost equally, so their probabilities do not differ much across reviews. Removing them reduces the vocabulary size. However, the effect is minimal because they appear across both classes.

The accuracy after disabling the default lowercase normalization is 0.81, which is 0.04 lower than the accuracy with Laplace smoothing. The vocabulary size increases and for example, *"brow"* and *"Brow"* will be two separate words. This means the word frequency of *"brow"* weakens because of the fragmentation.

3 Text classification with a bigram language model

3.1 Theory (2.5 pt)

Exercise 3.1.1

When f_i is a bigram (w_{i-1}, w_i), the positive class bigram probability is estimated as:

$$P(w_i \mid w_{i-1}, +) = \frac{C(w\ i-1, w\ i \mid +) + k}{C(w\ i-1 \mid +) + kV}$$

Where:

- $C(w_{i-1}, w_i \mid +)$ is the count of the bigram in the positive corpus
- $C(w_{i-1} \mid +)$ is the count of the first word of the bigram in the positive corpus
- V is the vocabulary size
- k is the Laplace smoothing constant

Exercise 3.1.2

Calculate the probability of "great movie" $\left(P(movie|great)\right)$ *and "familiar enough"* $(P(enough|familiar))$ *being positive (using Laplace smoothing).*

Using the previous formula, the probability for "great movie" $\left(P(movie|great)\right)$ being positive is:

- $C(\text{“}great\ movie\text{”} \mid +) = 1$

- $C(\text{“}great\text{”}, +) = 2$
- $V = 32$

$$\frac{C(\text{"}great\ movie\text{”} \mid +) + k}{C(\text{“}great\text{”} \mid +) + kV} = 1 + \frac{1}{2} + 32 = \frac{2}{34} = \frac{1}{17} = 0.0588$$

The probability for “familiar enough” being positive is:

- C(“familiar enough” | +) = 0
- $C(\text{“}familiar\text{”} \mid +) = 1$
- $V = 32$

$$\frac{C(\text{“}familiar\ enough\text{”} \mid +) + k}{C(\text{“}familiar\text{”} \mid +) + kV} = 0 + \frac{1}{1} + 32 = \frac{1}{33} = 0.0303$$

Exercise 3.1.3

Consider the review “the movie is intriguing”; what is the probability of this review being positive, using a bigram-based NB model trained on our corpus?

For “the movie is intriguing” the bigrams are “the movie”, “movie is”, “is intriguing”.
Following the formula used previously, the probabilities of these bigrams calculate to:

- $P(movie \mid the, +) = \frac{C(\text{“}the\ movie\text{”} \mid +) + k}{C(\text{“}the\text{”} \mid +) + 32} = 1 + \frac{1}{5} + 32 = \frac{2}{37}$
- $P(is \mid movie, +) = \frac{0+1}{2+32} = \frac{1}{34}$
- $P(intriguing \mid is, +) = \frac{1+1}{1+31} = \frac{2}{33}$

So, the probability of “the movie is intriguing” being positive would be: $P(the\ movie\ is\ intriguing \mid +) == P(movie \mid the, +) * P(is \mid movie, +) * P(intriguing \mid is, +) = \frac{2}{37} * \frac{1}{34} * \frac{2}{33} = \frac{2}{20757} = 0.00009635303$

3.2 Coding (2.5 pt)

Exercise 3.2.1

What is the accuracy of a model with n=2 on the test set? What could be the reason for the difference?

The accuracy of a model with n=2 on the test set is 89%. It is higher than the unigram classifier model, because the bigram model captures word order and context, which makes them able to distinguish sentiments. For example, “not good” vs “good not” or “very bad” vs “bad very”.

Exercise 3.2.2

Does setting n=3 or n=4 improve the accuracy? Why/why not?

Setting n=3 or n=4 decreases the accuracy of the model to 77% and 65% respectively. The reason for this is that the training dataset does not have that many higher order n-grams, compared to n=2.

Exercise 3.2.3

How could you improve the performance of the classifier? Give a couple of motivated suggestions (i.e. explaining why they should help); to get the full 2,5 points, implement them and see if they actually improve the performance on our dataset.

One suggestion would be to use TF-IDF weighting for bigrams, which downweighs terms common across documents and upweights distinctive terms, which often helps linear models like MultinomialNB to generalize better.

Another suggestion would be to use a mixed unigram and bigram model, because pure bigrams can be sparse and might miss single-word sentiment cues. Mixing the two models should balance coverage of strong single-word indicators with short context bigrams, typically improving performance.

After implementing the above-mentioned suggestions in code, the accuracy ended up being 85% and 84% respectively, which means no performance improvement in practice for our dataset.