**MSE\_AnTeDe\_Lab2 – Adrian Willi**

|  |  |
| --- | --- |
| **Questions** | **Answers** |
| Lab 2b We extract the features with CountVectorizer, removing the stop words automatically. Something goes wrong. What is it? | We didn't lemmatize our tokens and for this reason, the tokens bear and bears are carrying the same information. We should avoid that by lemmatizing our tokens. |
| Lab 2b In this example, Bern and Zurich get much higher weights than canton and capital. Why? | Because *canton* and *capital* are appearing in all documents which lead to a lower score while *Zurich* and *Bern* are each appearing twice in only one document. |
| Lab 2b Bonus question: how was the MNB able to classify It is the city of Zwingli., given that Zwingli doesn't appear in the training data? | Maybe the classifier used *city* to classify the sentence and not particularly *Zwingli.* In case there would be only unknown tokens in a sentence then the MNB would no be able to classify it. |
| Lab 2c How many training documents and how many test documents are there in the dataset? If the answer you get is 6 of each, you made a mistake. | 11’314 Training documents  7’532 Test documents |
| Lab 2c Now run mnb\_tfidf without removing the stopwords and analyze its performance. Why doesn't the performance drop as much as it did with CountVectorizer? | Words that appear frequently over different documents have low tf-idf scores because they don't contribute a lot information to a single sentence. For this reason, the score is better than with CountVectorizer. |
| Lab 2c Experiment with other classifiers (other than MNB) from scikit-learn. For instance, try Stochastic Gradient Descent. What's the best performance you can get using the default parametrization of the scikit-learn classifiers? | SGD with Tfidf Vectorizer: 0.71 (Accuracy)  Decision Tree with Tfidf: 0.445 |