Lab 2: Text Classification
Reflection Report

TDDE16 – Text Mining

Anonymous ID
82125

IDA
Linköping University

November 13, 2023

RQ 2.1: *Summarize the results of your experiments for Problem 2. Are your results "good" or "bad", and importantly, how do you determine that?*

The results of the experiments in Problem 2 show an overall accuracy of 43%, with quite low f1-scores for most of the parties. **M** and **S** show higher f1-scores, meaning that the model better handles the precision-recall-trade-off when classifying speeches from these two parties (f1-score show the harmonic mean of precision and recall). However, we have to remember that the data is unbalanced and both **M** and **S** have more speeches in both the training and test dataset (i.e. higher support).

Generally, we cannot expect high accuracy when doing this kind of task. It is easy to understand that many opinions will overlap between the different parties, and it would be a difficult task even for humans to classify speeches to specific parties. All in all the performance of the model is however "bad" since the accuracy is not equally spread across the parties, but instead the model is *OK* at classifying the big parties (which also had more data). The model is also a somewhat better at classifying the far left and right wing parties (probably since their opinions and use of language are more pronounced), while most of the smaller parties are harder for the model to classify (based on f1-score). Important to mention is however that we can accept a lower f1-score on this kind of task, since misclassification does not cause any danger (as it might do in for example a medical context, where we especially would not want false positives causing low precision).

Determining if a model is good or bad is therefore domain specific, and highly dependent on the context of the problem and what the model will be used for. One way to determine if a model is good is to compare with a baseline (for example, a most-frequent dummy classifier) and see if the implemented model is better or worse than the baseline (based on, for example accuracy and average f-score). Trying to improve in this model further should yield a good understanding of the limitations of different models and what can be considered as "good" for the specific task.

RQ 2.2: *In Problem 4, you implemented undersampling. How did your results change compared to Problem 2? How would "oversampling" have looked like for this task?*

Undersampling results in an equally distributed data set, where all parties now have an equal amount of speeches in the training data. The model trained on this data set yielded a slightly lower overall accuracy of 41%, however, the accuracy is much more "equally spread across all parties". That is, the model is now better at classifying the smaller parties, but slightly worse at classifying the bigger parties (**M** and **S**), based on f1-score. This seems reasonable, and while the overall accuracy is worse, this kind of model actually performs better since it is much more consistent across the different parties.

While you in undersampling remove samples from classes with many samples (majority classes) so that all classes are equal, oversampling instead duplicates samples from classes with lesser samples (minority classes).

RQ 2.3: *Why is it important to do a hyperparameter search before drawing conclusions about the performance of a model? Why do you think it is often not done, anyway? Why should you never tune hyperparameters on the test set?*

It is always important to do a hyperparameter search before drawing conclusions about the performance of a model, since without optimizing the hyperparameters, we cannot be sure that the model is performing at its optimum. When drawing conclusions and especially comparing different models, we of course want to compare the best possible performance of each model, and we cannot know what hyperparameter will perform the best without testing. Hyperparameter search can however be very computationally heavy and therefore also resource and time-consuming, which is why it is often not done.

The reason for not tuning the hyperparameters on the test set is that you will end up "leaking" the test set to the model, which will yield hyperparameters that will tune the model to perform the best it can *on that test set*. So, this will yield a non-generalized model which will perform poorly on new data. We always want to test the model on never before seen data, so that we can measure true generalization. This is why we include validation sets or use cross validation techniques in the hyperparameter tuning scheme.