DáilVote: Predicting results of Second Stage divisions in Dáil Éireann

Marja-Kristina Akinsha
Department of Computing, TU Dublin, Tallaght, Ireland
Supervisor: Jelena Vasic
X00170782@myTUDublin.ie

Introduction

My project predicts the results of votes in Dáil Éireann based on the features of the preceding debate. I have used the Houses of the Oireachtas APIs which provide data on parliamentary proceedings to generate a dataset which relates parliamentary speech to votes, which is restricted to Second Stage debates and votes during the 32nd Dáil (2016-2020). I have trained SVM and TensorFlow models on this dataset to predict how each TD will vote using features based on what they said in the preceding debate and/or other contextual features.

Data Extraction & Preprocessing

There have been similar studies performed using data from the UK parliament (Abercrombie and Batista-Navarro, 2020), and California Legislature (Budhwar, 2018). Based on these, I chose to generate the following feature sets: **bag-of-words** (unigram counts from speech text), **other speech features** (sentiment polarity, sentiment subjectivity, speech count, word count) and **non-speech features** (party, bill type, constituency, government/opposition).

In creating my dataset, I also applied the following recommendations from the literature to improve data relevance: focusing on the bill as a whole and excluding votes on amendments, eliminating votes other than Yes/No, and focusing on only one house/session of parliament.

I used a number of tools and libraries for data extraction and pre-processing. **Curl** was used to extract full debate transcripts and metadata on bills and members from the Oireachtas Open Data portal, and **BeautifulSoup** was used to filter for relevant XML tags. The data was transformed into CSV format, and votes were then correlated to the preceding speeches based on the bill reference in **pandas**. Non-English (primarily Irish) utterances were filtered out with **langid** for better accuracy in sentiment analysis. The data was cleaned using **NLTK** for sentiment analysis and bag-of-words transformation, and the resulting text was tokenized and stemmed. Finally, **TextBlob** was used for sentiment analysis (polarity and objectivity) and **scikit-learn** was used to create the bag-of-words representation.

Most commonly used words in bag-of-words

Model Development & Model Performance

Model Development

I ran some initial experiments training Support Vector Machines (SVM) and TensorFlow classifiers on all combinations of the generated feature sets. These classifiers were selected based on their successful use in prior studies.

Metric/Model	NAIVE	SVM_BOW	SVM_OS	SVM_OS_COUNT	SVM_OS_SENT	SVM_OF	**SVM_BOW_OF**	SVM_ALL
accuracy	0.7368	0.7348	0.6591	0.6591	0.7651	0.7575	**0.8258**	0.7803
precision	0.7368	0.7624	0.6591	0.6591	0.7651	0.7586	**0.875**	0.849
recall	1	0.875	1	1	1	0.9565	0.901	0.8738
specificity	0	0.45	0	0	0	0.30	**0.58**	0.45
F1 score	0.8484	0.8148	0.7945	0.7945	0.8669	0.8461	**0.8878**	0.8612

The best performing SVM model was trained on a combination of bag-of-words and other features. It achieved 82.6% accuracy and also scored highest in most other metrics. This was a clear improvement over the naive majority class classifier (always predicting a yes vote) which achieves 73.6% accuracy.

The initial Tensorflow experiments were performed on a model trained for 20 epochs, using the Adam optimizer, with two hidden layers of 24 nodes each.

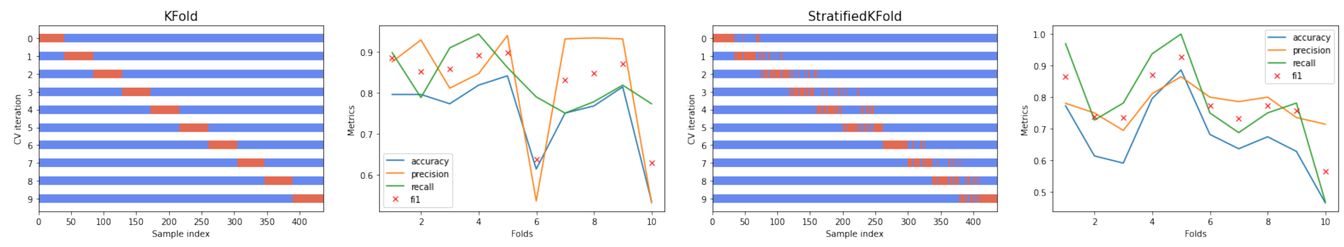
Metric/Model	TF_BOW	TF_OS	TF_OS_COUNT	TF_OS_SENT	**TF_OF**	TF_BOW_OF	TF_ALL
acc	1.000	0.6644	0.7260	0.7432	**0.8082**	1.000	1.000
val_acc	0.7655	0.6966	0.7103	0.7241	**0.8276**	0.8000	0.7931
loss	0.0029	0.7355	0.6282	0.5660	**0.3994**	0.0042	0.0034
val_loss	1.2700	0.6960	0.6691	0.5919	**0.4371**	0.9915	1.035

The results for TensorFlow were mixed. Most of the models did not have promising metrics compared to the naive classifier. The one exception was the model trained only on the other features set which had a validation accuracy of 82.8% which is similar to the best SVM model.

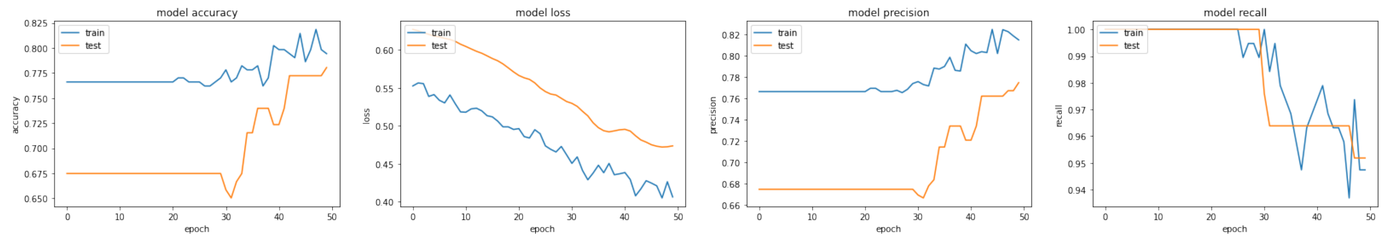
Model Performance

I tuned the most promising SVM and TensorFlow models further and evaluated their performance and generalizability.

The tuned SVM classifier had the following performance metrics: 82.6% accuracy, 87.5% precision, 87.4% recall, 58% specificity and F1 score of 0.8878. Running k-fold cross-validation on this model supports the model generalizing well, though the stratified k-fold cross-validation was a little less strong and could be an area to investigate further.



The tuned TensorFlow classifier had the following performance metrics before generalization: 79.4% accuracy, 81.4% precision, 94.7% recall, and F1 score of 0.8759. With generalization (adding dropout and Gaussian noise), the TensorFlow model had similar performance to a naive classifier. This suggests that it does not generalize as well as the SVM classifier.



Conclusions & Future Work

Overall, SVM was better suited for this task because it consistently generated better results than the naive classifier. Unlike the TensorFlow models it was not sensitive to small adjustments in hyperparameters, and it benefited from using a wider range of features which may have contributed to its stability.

Some suggested future research in this field would be running the model on a larger dataset. One approach would be to follow the suggestion of Kornilova et al (2018) by using bootstrapping techniques on the existing DáilVote dataset to provide additional data. Another approach would be to expand the dataset beyond the scope of the 32nd Dáil, although this may bring further challenges due to changes in the membership, constituencies, and parties.