UNIVERSITY OF PISA

DEPARTMENT OF COMPUTER SCIENCE

Human Language Technologies

Power Identification in Parliamentary Debates

Professor:

Lucia Passaro

Group 4:

Mirko Michele D'Angelo
Filippo Morelli
Filippo A. Sandoval Villarreal
Saul Urso

ACADEMIC YEAR 2023/2024

Contents

1 Introduction

Analyzing parliamentary speeches is a critical aspect of understanding modern political dynamics and discourse. Parliamentary speeches are a rich source of data that reflect the underlying ideologies[1], policy preferences[2], and rhetorical strategies of political actors[3]. Through the examination of these speeches, researchers can gain insights into the political landscape, uncovering the nuances of political communication and behavior.

In fact, identifying different perspectives in political debates is essential[4] to capture the full spectrum of viewpoints presented by political actors. This includes sentiment analysis to gauge the emotional tone of speeches[5, 6] and studying patterns of persuasion or manipulation[7].

Our initiative aimed to contribute to the advancement of the field, focusing on a task published by the Touché lab of CLEF 2024. Touché represents a series of scientific events and shared tasks to foster the development of technologies that support people in decision-making and opinion-forming and to improve our understanding of these processes. For our contribution, we focused on the following task:

Given a parliamentary speech in English from the Great Britain parliament, our goal is to identify whether the speaker's party is currently governing (coalition) or not (opposition).

2 Related Work

Due to the novelty of the task, which is from 2024, and the fact that the challenge is still ongoing, there are currently no state-of-the-art approaches specifically tailored to our dataset available to solve **power identification**.

However, other NLP tasks have already been addressed on similar datasets. For instance, **stance classification**, which involves classifying a user's stance in a debate (i.e., whether they are for or against the topic) [7], and **sentiment analysis** on parliamentary speeches [8], demonstrate the validity of using vector embeddings such as Word2Vec[9], FastText[10], or GloVe[11].

Other approaches, on the other hand, demonstrate how Recurrent Neural Networks (RNNs) can achieve good performance on similar datasets in argument mining[12]. Additionally, [13] illustrates how it is possible to extract estimations of **demographic traits from text**, including **party status** (coalition and opposition), which is relevant to our task.

Pretrained models have also proven to be a sensible approach in NLP for many tasks over the past few years [14, 15]. Leveraging models pretrained on similar domains [16] can be particularly advantageous, as they offer a strong starting point by incorporating a wealth of linguistic knowledge and contextual understanding.

3 The Data

The dataset for this task, is a labelled dataset derived from ParlaMint[17] multilingual comparable corpus of parliamentary debates, provided in tab-separated text files.

The focus of our report lies on the dataset containing speeches from Great Britain, despite the availability of diverse datasets in multiple languages for our task. This particular dataset comprises approximately 30,000 speeches. However, it lacks additional Development or Test sets. It's essential to note that our task restricts us to using only the text data for feeding into

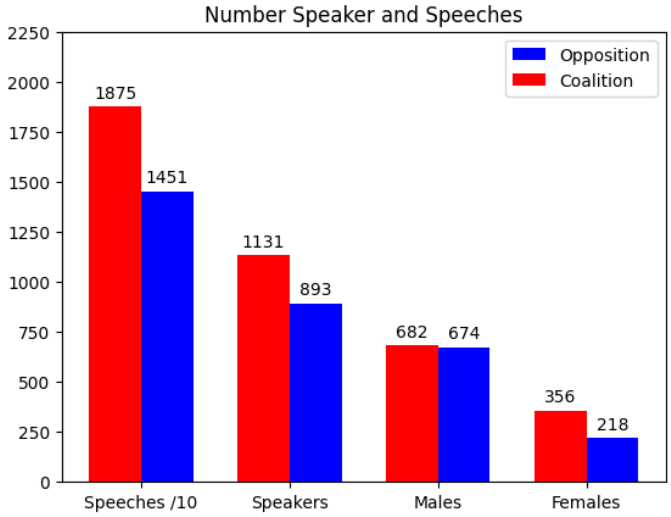

Figure 1: Dataset composition for each of the two classes, with the number of speeches divided by 10 for better visualization.

the models, excluding the other associated columns of "speaker" and "sex". Since the task is supervised, the dataset includes a "label" column with binary values: 0 indicates opposition, and 1 indicates coalition. For a more detailed understanding of the dataset structure, please refer to Appendix A Sec. 1.

The task, as established by the organizers, uses the **macro-averaged F1-score** as the **primary evaluation metric**.

Classes composition

Fig 1 reveals the dataset composition, which consists of 33,257 speeches, with 18,752 (56.4%) from coalition members and 14,505 (43.6%) from the opposition. There is a notable gender imbalance: 22,800 speeches are delivered by male speakers and 10,457 by female speakers. This imbalance is present in both classes, as both the coalition and the opposition have a majority of male speakers, with the coalition having more speakers overall than the opposition.

Word Frequency

An important aspect of our dataset analysis involves examining word frequencies. Using the "en_core_web_sm" model from SpaCy and removing stopwords, we identified the 10 most frequently used words, as shown in Fig. 2.

Words like "Government" and "People" are more prevalent in coalition speeches, while "Friend" and "Right" are more common in opposition speeches. Additionally, terms such as "Lords" and "Noble" frequently appear, highlighting topics of significant relevance in the British government.

This observation highlights a significant expectation: the corpus in question is highly specialized and contains terms not commonly used in everyday language (e.g., "hon. friend") which could potentially hinder the performance of pretrained models, depending on the corpus on which they were pretrained.

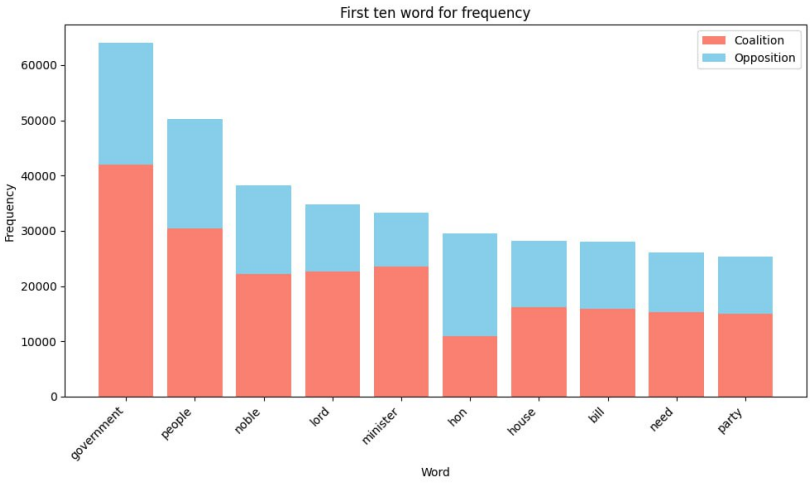

Figure 2: Top 10 most frequently used words in the dataset

4 Methodology

Initially, the dataset needed to be divided into three subsets: training, validation/development, and test sets. We adopted a hold-out method, reserving 10% of the dataset's size for the test set. The remaining data was also split in an hold-out fashion, allocating 20% of it for validation and the remaining 80% for training. To view the distribution of the speeches in the three splits we refer to Sec 1 of the appendix.

To address the task at hand, we initially established a basic baseline, consisting of a logistic regression model fed with TF-IDF document vectors, as outlined in Section 5.2. Subsequently, to improve upon and surpass this baseline performance, we explored several alternative approaches:

- Utilizing **vector embeddings** such as Word2Vec or FastText, either pretrained on external datasets or trained specifically on our dataset, to augment the logistic regression performance.
- Employing a **random forest classifier**, which could potentially yield superior performance if the logistic regression model proves insufficient.
- Training Bidirectional RNN (**BiRNN**) and **BiLSTM** models, more powerful models adept at capturing dependencies across the transcription.
- Leveraging a **pretrained transformer encoder** to evaluate its performance and effectiveness in our context, focusing on two models: RoBERTa [15] and POLITICS [16]. POLITICS is a variant of RoBERTa that has undergone specialized pretraining to optimize the model for ideology detection and stance detection in political contexts.

5 Experiments

5.1 Experimental setup

Given the collaborative nature of this report, different parts of the experiments were conducted using different machines. Table 1 presents the hardware specifications utilized for each experiment.

Table 1: Hardware used for the different kinds of models taken into consideration in the analysis

Experiment	**hardware**
Embeddings	Intel core I5 1135G7 @ 2.40GHz, RAM: 32GB
RNNs	Intel core I5 13600K @ 3.5Ghz, RAM: 16GB,GPU: rtx 4060 8GB vram
Random Forests	M2, RAM: 16GB, GPU: Apple 10-core
Pretrained models	Tesla P100-PCIE-16GB (up to 8 layers fine-tuned) Tesla V100S-PCIE-32GB (fine-tuning 10 and 12 layers)

5.2 Baseline

Our initial approach to tackle the problem involved using logistic regression with TF-IDF document vectors, applying sublinear TF to reduce the impact of very frequent words.

We conducted a grid search to identify the best hyperparameters for the logistic regression model. Specifically, we utilized L2 regularization, guided by the C hyperparameter, and experimented with different numbers of iterations (max_iter). The ranges of hyperparameters included in the grid search are shown in Table 2.

Table 2: The hyperparameters of the logistic regression, the solver was lbfg and the penalty was l2 and they were the same for every model

Hyperparameter	**values**
C	0.1, 1.0, 10, 100, 1000, 1500
Max_iter	100, 200, 500, 700

Table 3: Best performance of the TF-IDF over a logistic regression with all the values of the hyperparameters

Penalty	**C**	**solver**	**Max_iter**	**F1 score**	**Precision**	**Recall**
$l2$	1.0	$lbfgs$	100	0.780	0.783	0.869

5.3 Embeddings

Embeddings trained on our dataset

Starting with a logistic regression model, our initial step was to enhance its performance using vector embeddings. We trained Word2Vec[9] using the CBOW approach and FastText[10] on our dataset. Prior to training, we preprocessed the dataset samples using Gensim's "simple_preprocess" function. During experimentation, we varied the embedding sizes and context window sizes for both models. To derive the embedding for each transcription, we computed the mean of the embeddings of its constituent components (words for Word2Vec and subwords for FastText). The variations are summarized in Table 4.

Table 4: The values of the context window and vector size explored for wor2vec and fasttext. The values that are followed by a symbol were used only with the one with the same symbol(the value 50 was used with other values of the vector size, but the value 900 of the vector size was used only with 50

Hyperparameter	**values**
context_window	10, 20, 30 ,40, 50#, 70++, 100++, 200++, 500+, 1000*
vector_size	150, 300, 600, 900#, 1200++, 1800+, 2000*

To better assess the performance of the embeddings, we evaluated the logistic regression model using the same grid search for hyperparameters as described in Table 2. This approach allowed us to compare the effectiveness of different embeddings systematically.

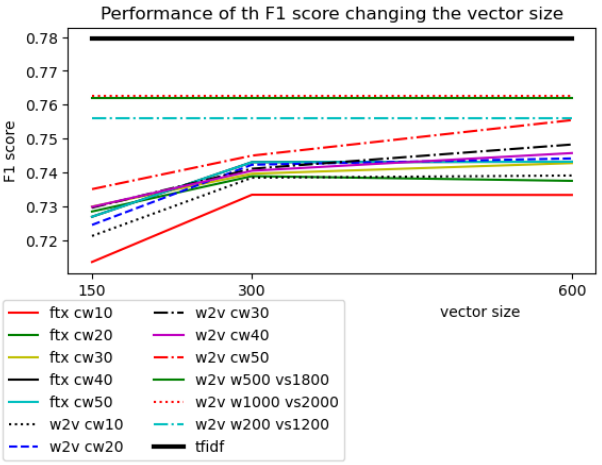

Figure 3: Behavior of the embeddings chainging the vector size

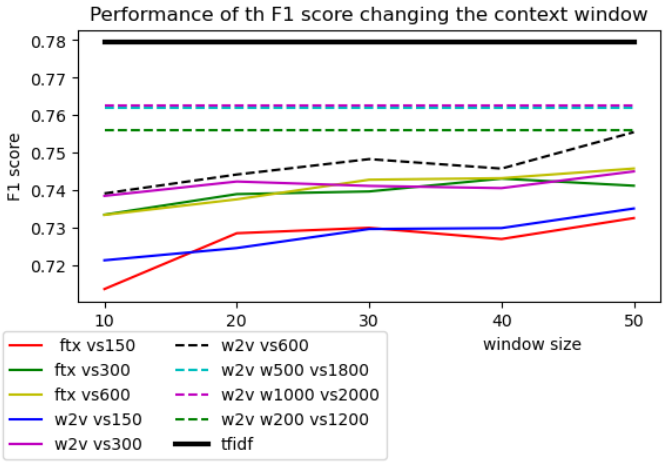

Figure 4: Behavior of the embeddings changing the size of the context window

Figs. 3 and 4 show, respectively, the effect of varying the embedding size and the context window size on the best F1 score obtained from the logistic regression grid search. It is evident that as the values of these parameters increase, the performance of the embeddings improves. This could be attributed to the fact that smaller context windows may not consider enough words to accurately infer the target, thus hindering performance, and it may also be attributed to the tendency of smaller embedding sizes to excessively compress the information of words. Notably, the embeddings show a lower F1-score value with respect to the baseline. The primary reason for this issue is the inherent difficulty in training effective static vector embeddings on small corpora. Such models require extensive exposure to each word to develop accurate embeddings. Consequently, rare words, in particular, do not appear frequently enough to form reliable vector representations.

Pretrained embeddings

To enhance our performance, since we encountered challenges in training vector embeddings that surpassed the baseline, we opted to utilize pretrained vector embeddings instead. We employed *w2v russ-corpora-300*, *google-news-300*[9], *glove-twitter-200*[11], and *glove-wiki-gigaword-300*[11].

Initially, we applied the pretrained embeddings without modification and assessed their performance using logistic regression. However, the results were suboptimal, with the best model, *google-news-300*, achieving a macro average F1 score of 72,6% as reported in Table 5.

Given the insufficient performance of the pretrained models, we decided to train Word2Vec and FastText models, starting from the pretrained versions. To accomplish this, we tokenized our training set with two different tokenizers: *punkt* from NLTK [18] and the pretrained WordPiece tokenizer from BERT [14]. We built the vocabulary for our Word2Vec and FastText models based on the tokenized training set and identified the words common to both our model and the pretrained model. We then initialized the embeddings of these common words in our model with their corresponding embeddings from the pretrained model, and then started training. Once trained we evaluate them using logistic regression.

We decided to intersect the common words between our model and the pretrained model because the pretrained models are trained on corpora that are very generic and not specific to our task. We made this decision to maintain the quality and contextual relevance of our embeddings, ensuring that we avoid incorporating unnecessary noisy embeddings.

Results and analyses

Next, we examine the results obtained using the different embeddings over a logistic regression applying the same grid search as in Table 2 . The results are reported in Table 5.

Table 5: The table reports the best performances of every model on logistic regression. The pretrained embeddings considered were used without modification from the 'Google-news' model and served as initialization for the common words between the pretrained models and our vocabulary, for the fine-tuned models (the ones containing 'FT' in their name).

Model	C	Max_iter	F1 score	Precision	Recall
baseline	1.0	100	**0.780**	**0.783**	**0.869**
$w2v\_cw1000\_vs2000$	10.0	500	**0.763**	0.778	0.832
ftx_cw600_vs50	100	500	0.747	0.761	0.830
$Google-news$	1500	500	0.726	0.743	0.818
$w2v\_FT\_Google\_punkt$	1000	100	**0.730**	**0.745**	**0.826**
$ftx\_FT\_Google\_punkt$	1500	700	0.727	0.743	0.820
$w2v\_FT\_Google\_bert\_based$	100	100	0.726	0.741	0.825
$ftx\_FT\_Google\_bert\_based$	10	200	0.727	0.744	0.819

Upon reviewing Table 5, it is evident that none of the computed embeddings surpass the baseline performance. The most notable result is observed with the Word2Vec embedding, utilizing a context window of 1000 and a vector size of 2000. Notably, among both fine-tuned and pretrained models, the Word2Vec model fine-tuned with the punkt tokenizer demonstrates the highest performance.

One reson for which the performance of the pretrained models fell short of expectations could be due to their embeddings being trained on corpora not specifically aligned with our task, thereby negatively impacting our results.

Moreover, while our "fine-tuned" models showed slight improvements over the pretrained models, they still did not meet our anticipated performance levels. This discrepancy can be

attributed to the fact that the pretrained embeddings were derived from corpora that were not specific for our task, leading to a limited overlap in vocabulary between our dataset and the pretrained embeddings. Consequently, this led to diminished performance as a result of both vocabulary mismatch and the inherent information loss during intersection. Additionally, the smaller vector sizes, as illustrated in Figure 3, contributed to suboptimal performance outcomes.

In summary, the attained performances failed to meet our initial expectations and did not surpass the baseline performance.

5.4 Random Forest

When we began training the Random Forest classifier, we used TF-IDF document vectors as input. However, despite the classifier's inherent complexity, it immediately became apparent that it could not achieve a satisfactory F-score, consistently remaining below 0.70. This outcome was unexpected, particularly given the model's complexity. To investigate the cause of this result, we explored **various preprocessing techniques**. We applied three main preprocessing techniques to address the unexpected results with the Random Forest classifier.

First, we **removed stopwords**, as these common words might not contribute meaningful information and could potentially introduce noise into the model.

Secondly, we **removed rare words** because they can significantly influence the learning process of machine learning models. We experimented with varying the threshold frequency for excluding words, testing with thresholds of 10, 100, and 1000 occurrences in the whole training set.

Finally, we experimented with **various tokenization pipelines** to assess their impact on the quality of tokens used in our model. Specifically, we tested three different Spacy pipelines: "en_core_web_sm," "en_core_web_md," and "en_core_web_lg."

Table 6: The values of the hyperparameters we explored for the Random Forest grid search. **NEst** is the number of decision tree. **Dep** is the maximum depth of each tree. **MinS** is the minimum number of samples required to split an interior node. **MinL** is the minimum number of samples that must be present in a leaf node

Hyperparameter	**Values**
*Num of estimators (**NEst**)*	1000, 2000
*Max tree depth (**Dep**)*	10, 25, 50
*Min samples to split (**MinS**)*	3, 5
*Min samples per leaf (**MinL**)*	2, 3

To evaluate the effectiveness of our approach, we conducted a grid search on the hyperparameters of the Random Forest classifier, as illustrated in Table 6.

The most significant results of the analysis for each of the three pipelines are presented in Table 7. As initially hypothesized, model performance improved with the exclusion of stopwords and infrequent words. Particularly with respect to rare words, it is noteworthy that, despite the expectation that excluding words with frequencies below 1000 might be excessive, given their relative commonality in a training set of approximately 27k samples, we observed improvements in the model F-score on the validation set.

Table 7: Best F-Score results for each pipeline. All the results are subjected to stopword removal. All the results have been obtained by removing the words with a frequency less than 1000.

Pipeline	NEst	Dep	MinS	MinL	Prec (Val)	Rec (Val)	F1 (Val)
sm	1000	50	3	2	**0.76**	**0.73**	**0.74**
md	2000	50	5	2	0.74	0.72	0.73
lg	2000	50	5	2	0.75	0.72	0.73

Regarding the Random Forest parameters, a maximum depth of 50 consistently yielded favorable results. Although this depth might suggest potential underfitting, this is not supported by the results: the model achieved a training F-score of 0.99, demonstrating its capacity to fully fit the training data.

An unexpected finding from our analysis was that the small (sm) pipeline of spaCy outperformed the other two pipelines. In fact, despite the larger and more complex nature of the other pipelines, which were trained on more extensive datasets compared to the subset used for the small pipeline, the smaller model yielded the best performance. This outcome is counterintuitive, given the expectation that the larger models would leverage their additional complexity and training data to achieve better generalization.

A potential explanation for this behavior is that none of the pipelines managed to surpass the baseline performance. This suggests that although there are performance differences among the pipelines, the overall model results remain suboptimal. Consequently, the small pipeline's relative success might be attributed to the general inability of all pipelines (and especially the Random Forest classifier) to achieve significant performance improvements beyond the baseline.

5.5 Recurrent Models

To further enhance the analysis, given our inability to surpass the baseline on the validation set, we experimented with Recurrent Models, which typically demonstrate strong performance in text analysis. Specifically, we focused on BiRNNs and BiLSTMs. Additionally, we experimented with initializing the LSTMs using pretrained embeddings. The models were implemented using PyTorch [19].

Training

For the **vocabulary**, when using the pretrained embeddings, we utilized only the intersection of the embedding vocabulary with our dataset vocabulary. When not using the pretrained embeddings, we included all the words in the vocabulary and we also experimented with cutting vocabulary size down to less than 10,000 words by cutting words with a frequency lower than 45. The text was tokenized using "en_core_web_sm", which had already proven to be a reasonable choice with Random Forests.

Training was conducted using **early stopping** with a patience of 5 epochs, a gradient clipping threshold of 1, and an embedding size of 128 or the size of the pretrained embeddings when used. The **loss** function we used is *binary cross entropy* which was optimized using the *Adam* algorithm[20]. Additionally during our experiments we also decided to use "**mean**

pooling": the method consists in using the mean across all the hidden states outputs of the RNN, both forward and backward for bidirectional models, as the output allowing the model to aggregate information from all time steps.

To classify the samples, the output is **fed into an MLP** with a sigmoidal activation function on all layers. We experimented with different numbers of layers, varying their sizes and using dropout on the intermediate layers for the MLP during our trials. The reason for varying the layers was to find an appropriate structure that was able to properly capture and generalize the features represented in the outputs created by the RNNs.

In Table 8, we reported the most significant results of our trials. The first trials were done with an higher learning rate and using BiRNNs which didn't manage to achieve good performance, probably due to their inability of capturing long range dependencies. Thus, in order to mitigate said problem, we resorted to LSTMs. After incorporating **dropout**, using mean pooling, removing lower frequency words, using more hidden neurons and layers, and optimizing the learning rate, we improved the baseline results, achieving an F1-score of 0.79 on the validation set. The increase in performance can be explained by the fact that cutting the vocabulary eliminated rarer words that could have been less informative and introduced noise in the representation, the increase of neurons and layers in the MLP improved the generalization capabilities but it lead to overfitting, thus we decided to employ dropout regularization and also added mean pooling which proved to be worthy and increased dramatically the performances.

Table 8: Most interesting results of our experiments with the RNNs with the relative model and hyperparameters configuration.

validation loss	training loss	F1 score	Precision	Recall	layers	hidden neurons	mean pooling	learning rate	model	vocab. size	dropout
0.9745	0.17695	**0.7503**	0.7525	0.7489	3	1024	*yes*	0.001	**BiLSTM-pretrained**	9944	0.2
0.7929	0.7100	0.5448	0.5460	0.5449	2	256	*no*	0.05	*BiRNN*	9944	0.0
43.1801	43.7790	0.3622	0.2840	0.5000	1	1024	*no*	0.05	*BiLSTM*	72276	0.0
56.6932	56.2583	0.3016	0.2159	0.5000	2	1024	*no*	0.05	*BiLSTM*	9944	0.0
0.8347	0.2096	**0.7962**	**0.8091**	**0.7915**	3	1024	*yes*	0.001	*BiLSTM*	9944	0.2

For the experiments with pretrained embeddings, we selected the configuration that reached the highest F1-score on the validation and tested it with the various pretrained embeddings used in section 5.3. Among these, the *glove-wiki-gigaword-300* embedding performed the best, achieving an F1-score of 0.75 on the validation set, indicated as **BILSTM-pretrained** on table 8. To view all the trials mentioned in this part we refer to Sec 3 of the appendix.

It appears that the pretrained embeddings employed in our model did not yield enhanced results, likely due to their derivation from a corpus that was too generic for the specialized nature of our task domain. This mismatch can lead to embeddings that are ill-suited for the specific vocabulary of our domain, thereby impeding the learning capabilities of the model.

From this batch of experiments we managed to observe that the various techiniques employed were useful to get a more robust model capable of generalizing, dropout allowed us to regularize the MLP parameters increasing its generalization capabilities and avoiding overfitting. As for the mean pooling it managed to achieve good results and mitigated the problem of long range dependecies. Finally using a reduced vocabulary also improved generalization capabilities by cutting of unnecessary and possibly noisy embeddings as shown by the results reported in A.3.

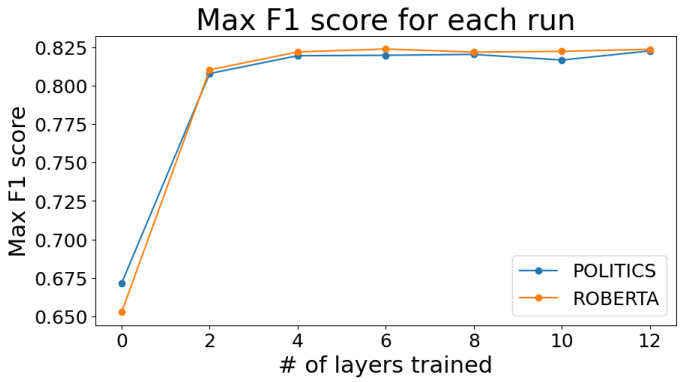

Figure 5: Best F1-score trend with increasing number of trained layers. 0 means only the final classifier was trained

Table 9: Best performance of the pretrained models on the Validation set

Model	Precision	Recall	F1 Score
POLITICS	**0.8392**	0.8168	0.8226
RoBERTa	0.8265	**0.8218**	**0.8237**

5.6 Pretrained Models

In our investigation, we also delved into the strategy of fine-tuning pre-existing models. Our attention was drawn to two specific models: RoBERTa[15] and POLITICS[16]. The POLITICS model, built upon RoBERTa through continued pretraining on English political news articles, was chosen especially for its better performances in tasks like ideology and stance detection.

We fine-tuned the models utilizing the Hugging Face libraries [21], employing CrossEntropy loss with default Trainer hyperparameters. These parameters include an AdamW optimizer with a linear decrease scheduling for learning rate (starting at 5e-5) a weight decay of 0, and default beta parameters[20, 22]. Evaluation of the models was conducted over the course of 5 epochs. The models were trained in minibatches, with a batch size of 32 samples.

Layer Training Effects

We aimed to explore the impact of varying the number of trained layers in both POLITICS and RoBERTa models, considering POLITICS' specialization in tasks similar to ours (political news articles versus parliamentary speeches). This analysis, ranging from training only the classifier to fine-tuning the entire model, was performed to evaluate the suitability of POLITICS for our task. Fig. 5 illustrates the F1-score across different numbers of trained layers (0 representing only the classifier and 12 indicating full model training). Each F1-score reflects the best performance achieved during the 5 epochs of model training.

The results indicate that fine-tuning the entire model achieves the best performance for both models studied. Notably, apart from the scenario where only the classifier is trained, RoBERTa consistently outperforms POLITICS in all trials.

Table 9 presents the two best models obtained in terms of F1-score on the validation set. Interestingly, RoBERTa outperforms POLITICS. This can be explained by examining Figure

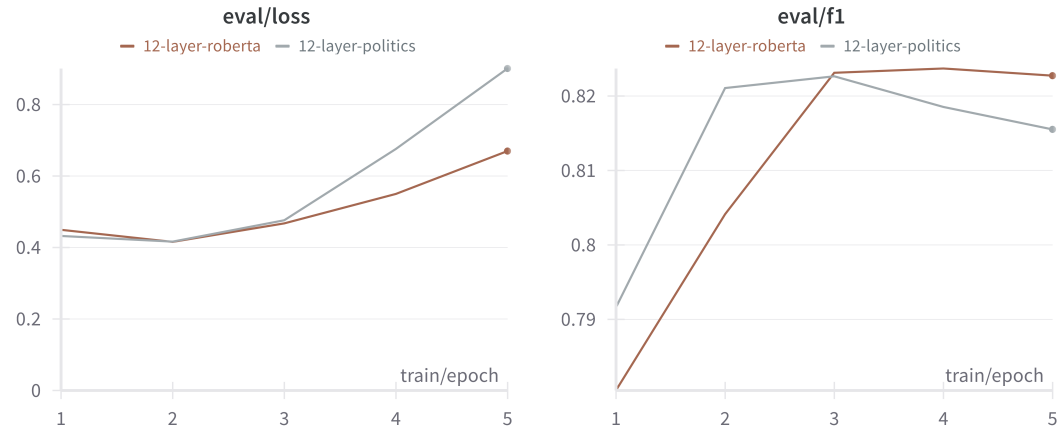

Figure 6: (Left) Validation loss for the best results of the two models studied over 5 epochs of training. (Right) F1-score of the two best models found over 5 epochs of training.

Table 10: Summary of the performances of the various models we examined during our project.

Model	Precision	Recall	F1 Score
baseline	0.771	**0.865**	0.769
$w2v\_cw1000\_vs2000$	0.763	0.830	0.750
RandomForest	0.747	0.720	0.723
RoBERTa	**0.845**	0.836	**0.839**
LSTMs	0.79	0.77	0.78

6. While the models achieve their maximum F1-score at either the 3rd or 4th epoch, their losses slightly increase by this point, indicating overfitting. RoBERTa likely performs better because its loss increases more slowly, resulting in a lower overall loss despite the overfitting (note however that, when reporting the validation and test results, we do not take the model obtained after 5 epochs, but the one that performs the best across the 5 epochs, which in this case is the model at epoch 4).

5.7 Test set results

The performance of the models examined in this report is summarized in Table 10. As anticipated from model selection outcomes, the pretrained model exhibits the highest F1-Score and precision. Notably, RoBERTa not only emerges as the best-performing model but also achieves better results on the test set compared to the validation set, which contrasts with the overall behavior of the other models, where test results tend to be slightly lower than those of the validation set.

Furthermore, the baseline model yields notable results, demonstrating competitive performance despite its simplicity. It achieves the highest macro-averaged recall among all the models evaluated.

To provide a clearer understanding of the models' behaviour, Figure 7 presents the confusion matrices for each model on the test set.

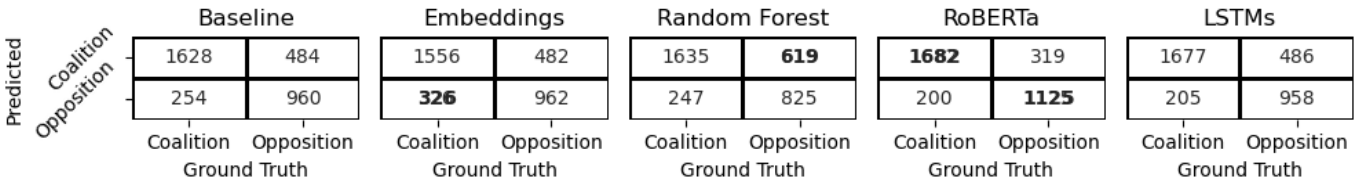

Figure 7: Confusion matrices for each of the five models runs on the test set

The confusion matrices reveal a key distinction: RoBERTa significantly outperforms the other models studied in correctly classifying opposition speeches, which represent the minority class. Although most models exhibit similar amount of correctly classified coalition speeches, the other models tend to **disproportionately favor the majority class**. This results in a range of approximately 500 to over 600 opposition speeches being misclassified as coalition, with the Random Forest model displaying the highest misclassification rate.

One reason for Random Forests displaying a significant bias towards the majority class may be that, by excluding terms with fewer than 1000 occurrences, we are implicitly filtering out words crucial for the opposition class. Since words more frequently appearing in coalition speeches are naturally overrepresented due to the dataset's class imbalance, this filtering disproportionately impacts the representation of opposition-related terms.

When examining the percentages, RoBERTa incorrectly classifies only 22% of opposition speeches. In contrast, the other models misclassify between approximately 33% and 42% of opposition speeches. Conversely, all models achieve similar accuracy in classifying coalition speeches correctly, ranging from approximately 89% for RoBERTa to around 82% for the embedding-based logistic regression model, which has the highest number of misclassified coalition samples (326).

This general behavior is anticipated due to the class distribution within the dataset. As outlined in Section 3, the coalition constitutes the majority class. Consequently, the models, being exposed to more samples from this class, develop a bias towards it, leading to the misclassification of opposition samples as coalition. To mitigate this effect, a straightforward and viable approach would be random oversampling or undersampling, which would address the inherent class imbalance in the dataset.

6 Conclusions

Analyzing parliamentary speeches is a valuable topic for NLP, with potential to enhance technologies that aid decision-making and opinion formation.

Our research aimed to classify British Parliament speeches as either from the governing party (coalition) or the opposition, starting with a simple baseline model and progressing to more complex ones, ultimately fine-tuning a pretrained model. Interestingly, our baseline model achieved notable results, indicating potential for further enhancement. We suggest exploring larger embedding sizes or expanding the context window, as well as testing embeddings with a Random Forest classifier for potentially better results. Additionally, improving the hyperparameters of the pretrained model and experimenting with LLMs might also improve performance, despite the limitations of decoder-only models. As a final suggestion, it is necessary to use the embeddings with the best text preprocessing, as we did for the Random Forest, to see if the performance improves.

In conclusion, studying parliamentary speeches through NLP offers significant technolog-

ical and societal benefits. While our research showed promising results, there is substantial scope for improvement. Future efforts should focus on optimizing embeddings, exploring alternative classifiers, and fine-tuning hyperparameters to develop more robust and accurate systems for classifying parliamentary speech and understanding political discourse.

Appendix A

Additional explanations

1 The Data

Dataset structure

Here we show with a toy example in Fig.A.1 the structure of the dataset we worked with.

```
id speaker sex text text_en label
gb01 spk1   F   First text. First text in English. 0
gb02 spk2   M   Second text. Second text in English. 1
gb03 spk3   M   Third text. Third text in English. 0
gb04 spk4   F   Fourth text. Fourth text in English. 1
gb06 spk5   M   Fifth text. Fifth text in English. 0
```

Figure A.1: An example of the dataset structure.

The fields of the dataset represent the following:

- **"id"** is a unique (arbitrary) ID for each text.
- **"speaker"** is a unique (arbitrary) ID for each speaker. There may be multiple speeches from the same speaker.
- **"sex"** is the (binary/biological) sex of the speaker. The values in this field can be Female and Male. Sometimes the value can be null, when it was not possible to find the sex of the speaker.
- **"text"** is the transcribed text of the parliamentary speech. Real examples may include line breaks, and other special sequences escaped or quoted.
- **"text_en"** is the automatic translation of the text to English. Useless to us since we are using solely the Great Britain parliament debates, which are in English.
- **"label"** is the binary/numeric label. For power identification 0 indicates opposition and 1 indicates coalition (or governing party).

Sentence length

Figure A.2 depicts the distribution of sentence lengths across each split of our dataset. The visualization reveals that a significant majority of speeches exhibit relatively short lengths,

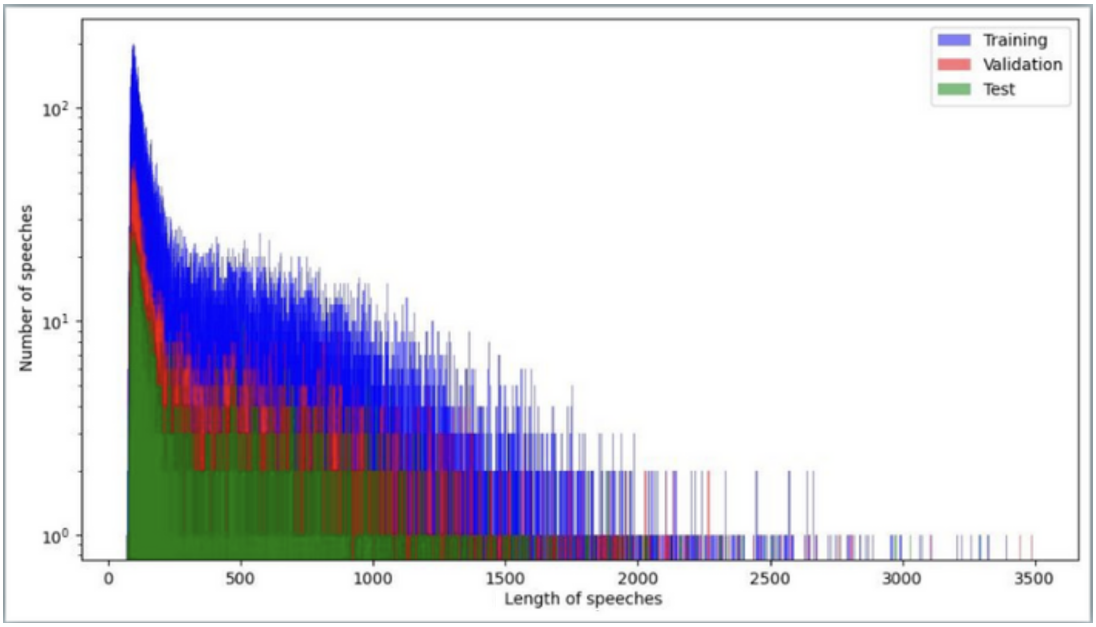

Figure A.2: Speech length histogram for the three splits of the dataset

as indicated by the prominent initial peak. Conversely, longer speeches are less frequent. It should be noted that while this distribution does not conform precisely to a power law, as evidenced by the shape of the curve, the trend is apparent.

Additionally, it is evident that the distributions across each split closely resemble one another, despite variations in their respective sizes. This consistency underscores our efforts to ensure a fair assessment of our models, minimizing any potential biases introduced during training.

Examining the lengths of speeches is crucial for informing our approach to training Recurrent Neural Networks. Both LSTMs and traditional RNNs encounter challenges in capturing long-term dependencies, a characteristic prevalent in our dataset. Thus, it is imperative to incorporate strategies that address these dependencies effectively when designing our models.

2 Embeddings

To evaluate our models we also decided to apply a word analogy test. In order to compute this score we use a test file for the word analogies, in particular we use the test of *question-word.txt*[23]. The table A.1 reports the score of the word analogy.

As observed, the pretrained models outperform our embeddings in the word analogy test. This disparity arises because pretrained models are trained on larger and more diverse corpora, enabling them to achieve higher accuracy. These extensive corpora encompass a broader range of contexts and writing styles, facilitating better learning of embeddings. In contrast, our dataset is smaller and more specialized compared to the corpora used in pretrained models, resulting in comparatively poorer performance.

An interesting observation is the superior generalization capabilities of FastText compared to Word2Vec. This outcome is predictable, given that FastText operates on subwords, allowing it to encompass a diverse vocabulary by combining embeddings of the q-grams it has learned.

Table A.1: The table reports the scores of word analogies for each model with dummy4unknown set to true and false. When dummy4unknown is set to true, the model replaces any unknown word with a "dummy" or a placeholder term to allow the model to continue the evaluation process even if it encounters words it doesn't know.

Model	Word analogy overall accuracy	Word analogy overall accuracy dummy4unknow
w2v_cw1000_vs2000	0.06	0.03
ftx_cw600_vs50	**0.44**	**0.25**
Google − news	**0.74**	**0.73**
w2v_FT_Google_punkt	0.37	0.24
ftx_FT_Google_punkt	0.36	0.23
w2v_FT_Google_bert_based	0.36	0.19
ftx_FT_Google_bert_based	0.49	0.26

3 Recurrent Cells

This chapter talks about the use of recurrent neural networks in our project, in particular we used Bidirectional Long-Short Term Memories (BiLSTM) and Bidirectional Recurrent Neural Networks (BiRNN). The models were implemented using PyTorch [19] and enhanced with various techniques such as gradient clipping, mean pooling, and dropout regularization.

Training

To train the models, a vocabulary with an embedding for each word was built using the *en_core_web_sm* tokenizer from Spacy. To aid generalization, words with less than 10,000 occurrencies were removed from the vocabulary during our experiments.

The training was done using an early stopping technique with a patience of 5 epochs and a maximum of 1000 epochs a gradient clipping threshold of 1 for the gradient and 128 recurrent cells for all models.

We used a *binary cross entropy* as an objective function and the *Adam* optimization algorithm. In order to read the neural embeddings we used feedforward neural network with multiple layers and the sigmoid activation function.

Bidirectional RNNs

Our first trials were done with BiRNNs, our approach was to keep a fixed size for the layers with 256 hidden neurons adding more layers to the MLP to observe the behavior of the model as shown in the table A.2. Adding more layers in this case result in worst performances.

Table A.2: This table reports the results of our experiments with the BiRNNs.

validation loss	training loss	F1 score	Precision	Recall	number of layers
0.7525	0.7441	0.5303	0.5300	0.5301	1
0.7929	0.7100	0.5448	0.5460	0.5449	2
0.6842	0.7026	0.3622	0.2840	0.5000	3
0.6888	0.7119	0.3622	0.2840	0.5000	4

Table A.3: This table reports the results of our experiments with the BiLSTMs

validation loss	training loss	F1 score	Precision	Recall	layers	hidden neurons	mean pooling	learning rate	vocabulary size	dropout
43.1801	43.7790	0.3622	0.2840	0.5000	1	1024	*no*	0.05	72276	0
56.6932	56.2583	0.3016	0.2159	0.5000	2	1024	*no*	0.05	9944	0
1.7257	0.1269	0.7798	0.7875	0.7765	2	1024	*yes*	0.001	9944	0.2
0.8347	0.2096	0.7962	0.8091	0.7915	3	1024	*yes*	0.001	9944	0.2
0.7123	0.6966	0.3622	0.2840	0.5000	4	1024	*yes*	0.001	9944	0.2
43.1460	43.7416	0.3622	0.2840	0.5000	4	2048	*yes*	0.05	9944	0
43.1460	43.7541	0.3622	0.2840	0.5000	3	4096	*no*	0.05	9944	0

For this part the vocabulary size has been kept fixed to 72776 for all the configurations of the model and a learning rate of 0.05.

Bidirectional Long-Short Term Memories

For the BiLSTMs we decided to add more neurons for the layers and also experimented with mean pooling and dropout. The results are shown on table A.3. As we can see increasing layers, cutting down vocabulary size,adjusting the learning rate and adding the dropout technique to the MLP resulted in an increase of performances.

Pretrained embeddings

Since the BiLSTMs showed better results we decided to use them and test their performance on the pretrained embeddings. In particular we tested *russ-corpora-300*,*google-news-300* and *glove-wiki-gigaword*. To load them we used gensim downloader api and used the them to build the vocabulary used by the models.

The training uses a similar setting to the previous one but with a learning rate of 0.001,a dropout with probability 0.2, 1024 hidden neurons for each layer and mean pooling for each model with a reduced vocabulary of 9944 words.

Table A.4: This table reports the results of our experiments on the pretrained embeddings

validation loss	training loss	F1 score	Precision	Recall	number of layers	embedding
0.9745	0.17695	0.7503	0.7525	0.7489	3	$glove-wiki-gigaword-300$
0.6845	0.7033	0.3622	0.2840	0.5	2	$glove-wiki-gigaword-300$
0.6926	0.7004	0.3622	0.2840	0.5	2	$russ-corpora-300$
0.6839	0.7037	0.3622	0.2840	0.5	2	$google-news-300$
0.6839	0.7037	0.3622	0.2840	0.5	3	$google-news-300$

As we can see the results are worse when compared to the best model withoout the pretrained embeddings, probably due the differences in the corpora use for the pretraining and the one used for our model.

Bibliography

[1] Shangbin Feng et al. *PAR: Political Actor Representation Learning with Social Context and Expert Knowledge*. 2022. arXiv: 2210.08362 [cs.CL].

[2] Daniel Schwarz, Denise Traber, and Kenneth Benoit. "Estimating the policy preferences of legislators in parliamentary systems: comparing speeches to votes". In: *Prepared for Presentation at the 71st Annual Conference of the Midwest Political Science Association*. 2013.

[3] Jennifer Jerit. "Issue framing and engagement: Rhetorical strategy in public policy debates". In: *Political Behavior* 30 (2008), pp. 1–24.

[4] David Vilares and Yulan He. "Detecting perspectives in political debates". In: *EMNLP 2017-Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2017, pp. 1573–1582.

[5] Zaher Salah. "Machine learning and sentiment analysis approaches for the analysis of Parliamentary debates". PhD thesis. University of Liverpool, 2014.

[6] Gavin Abercrombie and Riza Batista-Navarro. "Sentiment and position-taking analysis of parliamentary debates: a systematic literature review". In: *Journal of Computational Social Science* 3.1 (2020), pp. 245–270.

[7] Sakala Venkata Krishna Rohit and Navjyoti Singh. "Analysis of speeches in Indian parliamentary debates". In: *International Conference on Computational Linguistics and Intelligent Text Processing*. Springer. 2018, pp. 236–249.

[8] Elena Rudkowsky et al. "More than bags of words: Sentiment analysis with word embeddings". In: *Communication Methods and Measures* 12.2-3 (2018), pp. 140–157.

[9] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[10] Piotr Bojanowski et al. "Enriching word vectors with subword information". In: *Transactions of the association for computational linguistics* 5 (2017), pp. 135–146.

[11] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[12] Rory Duthie and Katarzyna Budzynska. "A Deep Modular RNN Approach for Ethos Mining." In: *IJCAI*. 2018, pp. 4041–4047.

[13] Huseyin Polat and Mesut Korpe. "Estimation of demographic traits of the deputies through parliamentary debates using machine learning". In: *Electronics* 11.15 (2022), p. 2374.

[14] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[15] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[16] Yujian Liu et al. *POLITICS: Pretraining with Same-story Article Comparison for Ideology Prediction and Stance Detection*. 2022. arXiv: 2205.00619 [cs.CL].

[17] Tomaž Erjavec et al. "The ParlaMint corpora of parliamentary proceedings". In: *Language resources and evaluation* 57.1 (2023), pp. 415–448.

[18] Edward Loper and Steven Bird. *NLTK: The Natural Language Toolkit*. 2002. arXiv: cs/0205028 [cs.CL].

[19] *Pytorch website*. URL: https://pytorch.org/.

[20] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[21] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL].

[22] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101.

[23] URL: https://raw.githubusercontent.com/nicholas-leonard/word2vec/master/questions-words.txt.