Traditional Methods Are Not Out of Fashion: Comparison of Models and Embedding Approaches for Parliamentary Text Classification

Hoang Ha Pham[*], **Xingming Li**[*], **Dariia Puhach**[*]
Uppsala University, Department of Linguistics and Philology
hoang-ha.pham.1833@student.uu.se
xingming.li.8883@student.uu.se
dariia.puhach.3065@student.uu.se

Abstract

In the current research, neural networks and LLMs dominate the NLP field, demonstrating competitive performance across language tasks. However, traditional machine learning methods can still offer high performance in more straightforward tasks like binary classification, while being more transparent and requiring minimal resources. To test this hypothesis, we evaluate traditional models, neural networks, and different text representations on a binary classification task using parliamentary debate transcripts. We found that traditional methods perform comparably to cutting-edge ones while requiring significantly fewer resources, suggesting they are well-suited for this problem. Additionally, the effectiveness of LLMs in this task varies across datasets.

1 Introduction

Recent years have seen neural network-based methods, particularly transformers (Vaswani et al., 2017), becoming the state-of-the-art (SOTA) approach for many natural language processing (NLP) tasks. While these models often outperform traditional machine learning methods, traditional models are still favored when interpretability is important (Li et al., 2022). In this study, we compare linear, ensemble, and neural network-based binary text classification models using parliamentary debate transcripts from the ParlaMint corpus. The task is to predict whether a statement is from the coalition or the opposition.

We also explore different feature types, from traditional word representations to modern pretrained word embeddings. Additionally, we assess the performance of these models against an off-the-shelf large language model (LLM) to determine whether its sentiment analysis capabilities can generalize to our classification task.

Our findings suggest that with the simple binary text classification task, traditional model architectures perform on par with or even better than more complex neural models while consuming much less computational resources.

2 Background

2.1 Dataset

The classification of speech as belonging to coalition or opposition parties was introduced by Touché[1] as a shared task, where data is sampled from ParlaMint, a large corpus of European parliamentary debates (Erjavec et al., 2022). The Touché dataset includes columns such as the speech transcript, its translation, and a label—0 for coalition and 1 for opposition. Party names mentioned by speakers were replaced with the placeholder `<PARTY>`.

2.2 Traditional vs. non-traditional methods

Traditional and non-traditional machine learning methods are frequently compared, as new models emphasize their accuracy improvements over the previous ones. In text classification, traditional models are praised for their interpretability and efficiency, while neural networks, ensemble methods, and LLMs are recognized for achieving high accuracy across a range of tasks (Li et al., 2022; Wu et al., 2020; Taneja and Vashishtha, 2022).

Traditional methods. In this work, we categorize linear statistical models as "traditional". Included in this group are: Linear Support Vector Classifiers (SVC), Support Vector Machines (SVM) with Stochastic Gradient Descent (SGD) training, and Logistic Regression. Linear SVC

[*]All authors contributed equally.

[1]https://touche.webis.de/clef24/touche24-web/ideology-and-power-identification-in-parliamentary-debates.html

seeks the optimal hyperplane that maximizes the margin between two classes, with the margin being a linear function (Hearst et al., 1998), and SVM is an extension of SVC that supports non-linear class boundaries. Logistic Regression models the probability of class membership using a logistic function. Stochastic Gradient Descent (SGD) is an optimization algorithm that updates model parameters incrementally using a gradient estimated from each sample at a time, or from a small subset of data (also called mini-batch). While linear models are often valued for their interpretability and low computational demands (Li et al., 2022), they oversimplify the true patterns in data due to their fixed probability function (Geman et al., 1992).

Ensemble methods. Random Forest and XGBoost are ensemble methods derived from decision trees. Random Forests, proposed by (Breiman, 2001), combine predictions from multiple decision trees to improve generalization. XGBoost, introduced by (Chen and Guestrin, 2016), combines decision trees iteratively and uses gradient descent to minimize prediction errors.

Neural networks. In contrast to traditional methods, neural networks (NNs) do not rely on a predefined probability function but learn to approximate it during training. While the bias-variance tradeoff is a common framework for understanding NNs (Geman et al., 1992), recent research indicates that proper optimization and increased model width can reduce both bias and variance (Neal et al., 2019). However, NNs are often criticized for being "black-box" models with high computational demands (Li et al., 2022). We use three types of NN architecture: a feed-forward NN, RNN, and LSTM. RNNs are designed to process sequential data, demonstrating high accuracy for certain text classification tasks (Amjad et al., 2019). LSTMs, an extension of RNNs, better capture long-range dependencies and address the vanishing gradient problem (Hochreiter, 1997).

LLMs are deep learning models trained on vast text data and can adapt to various downstream tasks without fine-tuning (Brown et al., 2020). They perform strongly in sentiment analysis (Upadhye, 2024), surpassing other methods even in zero-shot settings (Zhang et al., 2024). Llama family of models is no exception (Ji, 2024; Krugmann and Hartmann, 2024). We use open-source Llama3, which is a SOTA model (Dubey et al., 2024).

2.3 Feature representations

Traditionally, words are converted into numeric features using a feature engineering process. Bag-of-Words (BoW) with binary values or word count values and Term Frequency-Inverse Document Frequency (TF-IDF) are the most common methods to produce interpretable word representations (Jurafsky and Martin, 2009). Yet, the amount of information they can encode is limited. In contrast, modern approaches represent words as dense vectors in a high-dimensional embedding space, which can encode a staggering amount of information. Notable examples in this category include Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) which capture semantic relationships between words automatically, and contextualized embeddings like BERT (Devlin, 2018) and DistilBERT (Sanh, 2019) which account for variations in word meanings based on context.

3 Methodology

3.1 Classical models vs. Neural Networks

To evaluate the performance of traditional machine learning models, we train the following models: Linear Support Vector Classifier (LinearSVC), Logistic Regression, Support Vector Machine with Stochastic Gradient Descent (SVM-SGD), Random Forest Classifier, and two variations (10-iteration and 2000-iteration) of XGBoost. Except for XGBoost, all other models are implemented via the `sklearn` package, and we use the package's default hyperparameters. With XGBoost, we use `gbtree` booster with `binary:logistic` objective to output the prediction probability. All parameters for the tree boosters are the package's defaults. Due to a technical limitation, we cannot use of XGBoost's GPU acceleration and have to train the model on CPU.

For neural-network-based models, we use the `PyTorch` framework. We trained two variations of the feed-forward NN (with 1 and 3 hidden layers, layer size of 128 dimensions), and four LSTM networks of increasing complexity (two uni-directional with hidden sizes of 64 and 128; two bi-directional with hidden sizes 64 and 128). All variations have a learning rate of 0.01. For NN and LSTM models, we use 32 dimensions for the embedding layer and train for 10 epochs on GPU.

Tokenization for all models is done with a character n-gram tokenizer from `sklearn`.

3.2 Feature representations

To investigate how different features impact classification performance, we train a simple feed-forward NN using `PyTorch` framework with one hidden layer on different word embeddings. This ensures that performance variations, if any, are due to the feature representation rather than changes in model architecture. The models are trained for 10 epochs on GPU with a learning rate of 0.001.

Each embedding method is tested across multiple configurations, such as varying the max features for Binary BoW, TF-IDF, and GloVe, adjusting embedding dimensions for Word2Vec, and altering sequence lengths for DistilBERT. Binary BoW and TF-IDF are implemented via the `sklearn` package, and Word2Vec and GloVe embeddings are from the `gensim` package. With DistilBERT, we use `transformers` package from Hugging Face[2] to extract the embeddings.

3.3 Large Language Model

We used instruction-tuned Llama3 with 70B parameters (AI@Meta, 2024) through API calls on the platform groq[3]. We used this prompt for the zero-shot setting: "You are an expert in [country name] political life and parliamentary debates. Help me identify whether this speech was held by a coalition or opposition. First, reason about the text, then provide a final answer." We asked the model to provide reasoning, since in this way, generative models produce contextually aware answers. We set the temperature to 0 and increased it incrementally by 0.2 up to 5 times in case of JSON-generated errors in the requested output format. Since LLMs can suffer from data contamination—specifically, leakage from benchmarks (Li and Flanigan, 2024)—we followed the method outlined by Li and Flanigan (2024), attempting to retrieve ParlaMint data from Llama3. However, our prompts only resulted in hallucinations.

4 Experiments and Results

4.1 Experiments setup

To compare classical, ensemble, and neural-network-based models and various features, we trained several models on the same **GB** dataset containing parliamentary debates from Great Britain. The dataset includes 33,257 statements in English with a balanced class distribution. The reported metrics (Accuracy, F1-score, AUC, and training time) are averaged over five folds of cross-validation.

In the LLM experiment, we tested data from various countries, including Great Britain, Finland, France, Czech Republic, Ukraine, and Croatia, with all samples in their original languages. However, due to platform rate limits, the reported metrics are based on a single experiment.

All experiments were run under CentOS Linux release 7.9.2009 (64 bit) on Intel Xeon E5-2660 2.2 GHz processors, with 16 processors of 16 cores each, with 128 GB of RAM and 20 MB L3 cache. `PyTorch` models are trained on Tesla T4 GPU, and all others are trained on CPU only.

Model	Acc.	F1	AUC	Time
LinearSVC	0.785	0.817	0.858	1.30
LogReg	0.787	0.820	0.861	0.38
SVM-SGD	0.780	0.819	0.856	**0.09**
RandomForest	0.742	0.788	0.810	7.78
XGB-10	0.751	0.793	0.819	1.29
XGB-2000	**0.791**	**0.823**	**0.866**	77.29
NN-1L	0.759	0.788	0.835	1.26*
NN-3L	0.760	0.786	0.837	1.39*
RNN-1D-64	0.691	0.775	0.797	26.76*
RNN-2D-64	0.703	0.781	0.786	35.98*
RNN-1D-128	0.713	0.767	0.780	31.26*
RNN-2D-128	0.690	0.766	0.772	42.90*

Table 1: Comparison of model performance and average training time (minutes) per CV fold. The boldface indicates the best performance on each metric. Training time of models trained on GPU is marked with *.

4.2 Comparison of models architecture

The results of this experiment are reported in Table 1. XGBoost-2000 iter performs best overall, but its training time without GPU exceeds that of feed-forward and LSTM networks. Out-of-the-box Random Forest and XGBoost (10 iter) perform similarly. The performance gap between the two XGBoost models highlights the importance of hyper-parameter tuning for ensemble models.

On the other hand, linear models perform remarkably well across all reported metrics, and

[2]https://huggingface.co

[3]https://groq.com/

Feature	Acc.	F1	AUC	Time
BoW-5k	0.748	0.779	0.827	**0.21**
BoW-10k	0.753	0.788	**0.830**	0.22
BoW-20k	0.751	0.779	**0.830**	0.24
TF-IDF-5k	**0.754**	0.784	0.828	0.22
TF-IDF-10k	**0.754**	0.784	0.829	0.23
TF-IDF-20k	0.749	0.778	0.825	0.27
W2V-100	0.719	0.755	0.794	2.99
W2V-200	0.740	**0.789**	0.814	3.31
GloVe-100	0.658	0.729	0.708	3.41
GloVe-200	0.691	0.750	0.751	3.41
DBERT-64	0.674	0.740	0.732	9.95
DBERT-256	0.722	0.779	0.798	10.08

Table 2: Model performance across feature configurations, with average embedding time (minutes) per CV fold. Configuration values denote max features for Binary BoW, TF-IDF, and GloVe, dimensions for Word2Vec, and max lengths for DistilBERT. Boldface marks the best result in a metric.

their training times are much lower than those of ensemble and neural network-based models. Notably, Logistic Regression performs the best among the linear model, with Accuracy, F1, and AUC scores on par with XGBoost-2000.

Feed-forward NN performs slightly better with more hidden layers. In the case of RNNs, a larger hidden layer size seems to help more with performance than bi-directionality. The most complex model (RNN-2D-128) performs the worst.

4.3 Comparison of feature representations

Results in Table 2 reveal that complex features like GloVe and DistilBERT do not consistently outperform simpler methods. Both Binary BoW and TF-IDF provide stable performance, with TF-IDF achieving the highest accuracy (0.754) using either 5000 or 10000 features, and Binary BoW demonstrating exceptional speed at 0.21 minutes. Word2Vec yields a slightly lower accuracy but a higher F1-score when configured with 200 dimensions, though it takes around 3 minutes for embedding. GloVe, using 300-dimensional embeddings, shows lower performance due to limited features. DistilBERT-256 with rich contextual information gives F1 a boost but requires high embedding time.

4.4 Large Language Model

Table 3 demonstrates Llama3's zero-shot classification performance across multiple languages. High-resource languages such as English and French show relatively high accuracy, while low-resource languages like Croatian (HR) perform poorly (with Finnish diverting from the trend). F1-score drops significantly compared to accuracy when the dataset is imbalanced.

Dataset	Coalition%	Accuracy	F1
GB	44%	**0.78**	**0.76**
FI	55%	0.75	0.72
FR	63%	0.74	0.63
CZ	48%	0.70	0.70
UA	69%	0.68	0.59
HR	60%	0.48	0.42

Table 3: Llama3 performance using zero-shot settings for untranslated languages, sorted descending by accuracy. The boldface indicates the best performance on each metric.

5 Conclusion

In this study, we demonstrate that traditional machine-learning methods remain a viable choice for straightforward text classification tasks. Specifically, linear classifiers using Support Vectors can perform on par with complex ensemble or neural network models while requiring only a fraction of training time. Likewise, within neural network setups, simple Bag-of-Words and TF-IDF features perform better than dense word vectors while demanding significantly less computational time. LLM (LLama3 in particular) under a zero-shot setting, while can predict better than chance, still underperforms compared to simpler models. These results suggest that carefully selecting models and features considering the task's characteristics is more effective than automatically opting for large, complex models.

6 Limitation

Due to time and resource limitations, our selection of model architectures—both traditional and neural network-based—was restricted, and we did not conduct parameter tuning. This could contribute to the lower performance of neural network-based models in this classification task. In the first and second experiments, we only evaluate models on a single language (English) due to the added complexity of processing other languages. Future research could expand on this by testing other languages with varied class distributions and data

sizes, or conducting hyperparameter tuning to enhance model performance.

References

AI@Meta. 2024. Llama 3 model card.

Maaz Amjad, Alexander Gelbukh, Ilia Voronkov, and Anna Saenko. 2019. Comparison of text classification methods using deep learning neural networks. In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 438–450. Springer.

Leo Breiman. 2001. Random forests. *Mach. Learn.*, 45(1):5–32.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, and Filip Radenovic et al. 2024. The llama 3 herd of models.

Tomaž Erjavec, Maciej Ogrodniczuk, Petya Osenova, Nikola Ljubešić, Kiril Simov, Andrej Pančur, Michał Rudolf, Matyáš Kopp, Starkaur Barkarson, Steinór Steingrímsson, Çağrı Çöltekin, Jesse de Does, Katrien Depuydt, Tommaso Agnoloni, Giulia Venturi, Maria Calzada Perez, Luciana Macedo, Costanza Navarretta, Giancarlo Luxardo, and Darja Fiser. 2022. The parlamint corpora of parliamentary proceedings. *Language Resources and Evaluation*, 57.

Stuart Geman, Elie Bienenstock, and René Doursat. 1992. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

S Hochreiter. 1997. Long short-term memory. *Neural Computation MIT-Press*.

Peiqi Ji. 2024. Text sentiment analysis based on llama models. In *Conference on Computer Graphics, Artificial Intelligence, and Data Processing*.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Jan Ole Krugmann and Jochen Hartmann. 2024. Sentiment analysis in the age of generative ai. *Customer Needs and Solutions*, 11(1):3.

Changmao Li and Jeffrey Flanigan. 2024. Task contamination: Language models may not be few-shot anymore. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):18471–18480.

Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S Yu, and Lifang He. 2022. A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(2):1–41.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. 2019. A modern take on the bias-variance tradeoff in neural networks.

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

V Sanh. 2019. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Khushboo Taneja and Jyoti Vashishtha. 2022. Comparison of transfer learning and traditional machine learning approach for text classification. In *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 195–200.

Akshata Upadhye. 2024. Sentiment analysis using large language models: Methodologies, applications, and challenges. *International Journal of Computer Applications*, 186(20):30–34.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Hongping Wu, Yuling Liu, and Jingwen Wang. 2020. Review of text classification methods on deep learning. *Computers, Materials & Continua*, 63(3).

Ting Zhang, Ivana Clairine Irsan, Ferdian Thung, and David Lo. 2024. Revisiting sentiment analysis for software engineering in the era of large language models. *ACM Trans. Softw. Eng. Methodol.* Just Accepted.