Unveiling Insights from German Election Manifestos: A Natural Language Processing Approach"

Jan Mühlnikel[1], *Niklas* Scholz[1], *Luca* Mohr[1], and *Christian* Schmid[1]

[1] *Baden-Württemberg Cooperative State University (DHBW), 68163 Mannheim, Germany*

Abstract. This paper provides a detailed examination of political manifestos through the application of Natural Language Processing (NLP) techniques. The main objective is to extract and interpret core themes, ideologies, and sentiments. The methodology involves text extraction for manifesto retrieval, similarity analysis for document comparison, and sentiment analysis to evaluate the emotional tone and strategies of political parties. Furthermore, the manifestos are analysed using top words and topic modelling to identify key focus areas. The combination of Python for backend and ReactJS for frontend development ensures efficient processing and accessibility.

The findings provide important insights into political discourse, revealing the nuances and strategies employed in party manifestos. The use of NLP techniques offers a new approach to comprehending political narratives, which is advantageous for analysts, linguists, and individuals interested in political communication.

Introduction

Politics is a dynamic landscape where political manifestos serve as the basis for parties to present their ideas and ideologies. It is therefore crucial to understand these manifestos in detail. By analysing them with natural language processing methods, this crucial information can be revealed.Natural Language Processing (NLP) is a set of computer techniques designed to understand and work with human language in a way that mimics human understanding. The aim is to enable computers to process language in a way that resembles human understanding, facilitating a wide range of tasks and applications [1]. In order to uncover this information, several NLP methods are used in this project. Text extraction is used to extract the text from the manifestos provided by the parties. Text similarity is used to compare the different party manifestos. Sentiment analysis is used to analyse the extent to which a party uses its manifesto to influence opinion in a desired way. Furthermore, top words and topics are extracted from the manifestos to analyse where the parties focus on.

1 Related Work

[2] This paper by Paritosh D. Katre demonstrates the effectiveness of NLP in analysing political speech transcripts from different sources. Using Python libraries, it employs visualisations such as Lexical Dispersion Plot and WordCloud to efficiently perform Political Discourse Analysis (PDA). The NLP-based approach provides valuable insights into common themes across a large number of transcripts, offering a faster and less complex alternative to manual linguistic methods.

[3] This research introduces TSSSM (Tweet Semantic and Syntactic Similarity Measure), a method for measuring semantic and syntactic similarity in political tweets. Using word embeddings for semantic analysis and incorporating syntactic features, TSSSM outperforms existing measures. Experiments on political tweet datasets demonstrate its effectiveness, suggesting its potential for identifying and analyzing political narratives, with promising results on diverse test datasets.

[4] This study, by S. Orellana and H. Bisgin, uses NLP to analyse New Zealand political party manifestos (1987-2017). NLP techniques, including document similarity, topic modelling and sentiment analysis, provide insights into party evolution and policy attitudes. Findings suggest valuable insights, but recommend further refinement of NLP applications to political texts.

[5] This paper investigates the comparison of campaign proposals in the 2021 Ecuadorian presidential election using NLP, specifically the Doc2Vec algorithm. Using Spanish Wikipedia articles, two neural network models were applied to classify manifesto paragraphs into seven domains. The results show that Doc2Vec provides reasonable results.

Our research is characterised by the application of NLP analysis to German election manifestos. By employing various NLP techniques, we extract valuable insights from political party manifestos. In particular, we have developed a weighted cosine similarity score that integrates three different text representation schemes. To

increase accessibility and engagement, we also created a user-friendly website. This platform enables individuals to independently analyse relevant information from different German political parties, providing a comprehensive understanding of their respective manifestos.

2 Methodology & Theoretical Background

2.1 Text extraction

Extracting text from PDF files depends on the type of PDF file. These types can be categorised as follows.

Digitally created PDF files: These PDF files have been digitally created so that the text can be searched and copied.

Scanned PDF files: These are documents that have been scanned as images and saved as PDF files. The text cannot be copied as with the first type.

oCRed files: When the documents were scanned, the text was recognised using OCR and stored in the metadata of the PDF file, so the document is searchable and the text can be copied, but the file still looks like it was scanned [6].

In our case, since the election manifestos fall into the category of digitally created PDFs, we can take advantage of this feature and avoid using OCR and instead use text extraction software such as pypdf. These can use more information than the visual representation of the PDF, such as information about the encoding, font and typical character spacing. This advantage comes into play, for example, when identifying easily confused characters such as "oO0ö", as the text extraction software reads the electronically stored characters [6].

2.2 Text similarity

In general the similarity of two text is defined as the commonality between some text A and some text B. The more commonality they share the more similar two texts are [7]. Before text similarity can be measured, it is useful to preprocess the document. Text preprocessing is divided into several steps. First of all the text has to be tokenised. This means that ... Once the text has been tokenised, it is possible to eliminate stop words. This should make the text look less heavy and remove the dimensionality of the term space, making similarity analysis easier. Stop words include articles, prepositions and pro-nouns [8]. Additionally, lemmatisation needs to be applied. Lemmatisation is a linguistic technique that involves examining words through vocabulary and morphological analysis with the aim of removing inflectional endings and returning them to their base or dictionary form. This process ensures correct usage by distinguishing between verbs and nouns. In addition, lemmatisation facilitates synonym matching through the use of a thesaurus, thereby improving search results. For example, a query for 'hot' would also return results for 'warm', and a search for 'car' would include both 'cars' and 'automobiles' [9]. In order to analyse text similarity, you have to analyse string-based similarity, which works on and measures string sequences and character composition. To compare two texts, it was decided to look not only at the whole text, but also at the terms that appear in the text. This method is called term-based similarity. With this method, the terms in a text are represented as vectors, and with these vectors it is then possible to compare them. There are two commonly used similarity measures.

Manhattan Distance: The grid-like path that would have been travelled between two data points.

Cosine similarity: A measure between two vectors of an inner product space that measures the cosine of the angle between them [10].

Before you can calculate the similarity of the two texts, you first have to choose a representation scheme to vectorise the texts. In this work, three different commonly used representation schemes have been considered.

Bag of Words (BoW): BoW is based on creating a vocabulary from unique words in the text and representing the text as a vector based on word frequency, with the vector size equal to the vocabulary size. This approach captures the presence and frequency of words and provides a basic representation of the text.

TF-IDF: Term Frequency (TF) measures how often a word occurs in a review, expressed as the ratio of its occurrences to the total number of words in that review. Inverse Document Frequency (IDF) takes into account the importance of the word in the entire review corpus, with higher values for rarer words. The combination of TF and IDF in the TF-IDF equation emphasises both rare and common words, making it a widely used technique for creating meaningful word representations in text.

Word2Vec Operating as a neural network model, it grasps connections between words of similar meanings, like 'tasty' and 'delicious,' positioning them closely in a vector space. The internal mechanisms, resembling a deep learning black box, autonomously construct relationships among words from raw text, crafting a potent representation, particularly in larger dimensions.

Doc2Vec: Doc2Vec, an extension derived from Word2Vec, is an NLP vectorizer designed to represent entire documents as vectors, regardless of their length, with a primary focus on capturing sentence similarity [11].

Having chosen a representation scheme, it is now possible to calculate the similarity of the different party manifestos. To do this, we apply the previous similarity methods to the chosen vectorial representation scheme.

To apply similarity analysis to election manifestos, we decided to use a pre-trained model. Such pre-trained models are also called transformer models. These generic models are already pre-trained on a lot of text and therefore do not need to be trained manually [12]. For our use case, we chose to use the 'de_core_news_sm' model, which is already included in the NLP space library. This model has the advantage that it is already trained on German text data and is therefore perfect for analysing German election manifestos.

2.3 Text sentiment

Sentiment analysis aims to extract opinion from text, known as opinion mining. The goal is to learn about the public perception of a text. To do this, sentiment analysis uses natural language processing (NLP) [13]. Sentiment analysis can be applied at three different levels, with the first two levels representing the standard methods and the third a more advanced method:
Document level: At this level, the whole document is considered as a unit of information.
Sentence level: At sentence level, sentiment analysis classifies a sentiment for each sentence in a document.
Aspect level: This does not look at language constructs such as sentences or documents. Instead, this level focuses on specific aspects of entities. This means that different opinion holders can have different opinions on the same topic, and therefore this fact has to be taken into account in the analysis [14].
In this project, only the first two levels were considered and sentiment analysis was applied at sentence level. If sentiment analysis is used at sentence level, the output of the analysis would be a 'positive', 'negative' or 'neutral' sentiment for the input sentence. These sentiments are then aggregated to calculate the sentiment of the whole document or used individually [15].
We have chosen to use the 'germansentiment' model to perform sentiment analysis on election manifestos. This model is based on the Google BERT structure and has been trained on a large dataset of 1.834 million German language samples. The decision to use this model was based on its specialised design for detecting sentiment in German text, which ensures a more comprehensive and accurate analysis [16].

2.4 Top words

Visualising the most frequent words in election manifestos is a useful tool for highlighting the main themes and priorities expressed in these documents. This representation offers a clear and accessible way to identify the key issues and focus areas that political parties and candidates are emphasising in their manifestos. However, it is limited in its ability to provide a detailed analysis of the topics covered and only provides a superficial visualisation. To improve visualization, our project includes topic modeling.

2.5 Topic Modeling

This part of our project applies Natural Language Processing (NLP) techniques with the spaCy framework to categorise German texts into specific themes. Environment, Education, Health, and Economy. The de-core-news-lg model, a part of spaCy known for its effectiveness in processing German language data through neural network-based word vectors, is used for this purpose [17]. One key methodology employed is K-Fold Cross-Validation, a robust technique for assessing model performance in machine learning [18]. This involves dividing the dataset into K subsets and using each subset in turn for testing while the others serve for training, ensuring a comprehensive evaluation.
In addition, we use minibatch processing, a technique that improves training efficiency by reducing memory usage and accelerating convergence. This is particularly important in machine learning optimizations [19].
This combination of spaCy's language processing capabilities with K-Fold Cross-Validation and minibatch processing offers a reliable approach for automated text categorization, showcasing potential applications in diverse areas of text analysis.

3 Architecture

The architecture of software projects plays a crucial role in the development of efficient and reliable applications. A common concept in software development is to divide a project into a backend and a frontend.
We have adopted this architectural structure in our current project. It consists of a Python backend and a ReactJS frontend. These two components work closely together to ensure a seamless user experience.

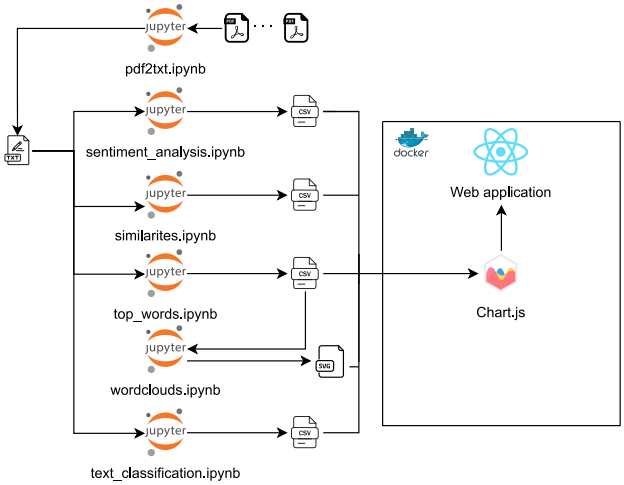

Figure 1. Architecture of the whole process

3.1 Backend

The Python backend is responsible for running Jupyter notebooks. These notebooks contain all the logic for the creation of analyses that will be executed on the host system. The results of these calculations are stored in two main directories. First, in the python directory, where they are stored in a structured form as CSV files. Second, in the ReactJS application's src directory, where they are made available to the frontend.

3.2 Frontend

The ReactJS front-end application runs in a Docker container, which has the advantage of not requiring a separate JavaScript runtime such as Node.js on the host operating system. This containerisation greatly simplifies the

Table 1. Similarity results for 'CDU & CSU' party

	BoW	TF-IDF	Doc2Vec	Weighted
SPD	81%	77%	17%	58%
Grüne	81%	77%	24%	61%
AFD	51%	39%	35%	42%
Linke	64%	58%	30%	51%
FDP	55%	44%	28%	42%

deployment and scaling of the front-end application.
In the front-end application, the exported results are read as dataframes using the Danfo.js library. This allows the data to be efficiently processed and analysed within the application. The data is dynamically visualised using Chart.js, a powerful library for creating interactive charts and graphs.

4 Results & Discussion

To calculate text similarity, three different representation schemes were compared. The similarity between all election manifestos was then measured using cosine similarity. Furthermore, an attempt was made to use all three representation schemes with their cosine similarities and to calculate a similar weighted cosine similarity over all representation schemes. Table 1 shows the similarity results obtained for the party 'CDU & CSU'. In analyzing the results, we have determined that, for our specific use case, measuring similarities using the 'TF-IDF' model proves to be the most effective. This choice aligns closely with our expectations for similarity measurement. Upon evaluating the 'Doc2Vec' model, we encountered a notable issue where the 'AFD' party appeared to be the most similar to nearly every other party, which did not accurately reflect the underlying truth. Subsequently, we compared the two standard models and ultimately opted for 'TF-IDF' due to its more advanced text representation. Additionally, we considered the weighted model. Although it produced logically reasonable results, the similarity values ranged narrowly, making them closely clustered compared to the other models. This was attributed to the fact that similarities, especially in the 'Doc2Vec' model, significantly varied with different representation schemes. In conclusion, the 'TF-IDF' model emerged as the most logical and reasonable choice, offering a comprehensive representation of text similarities in our context. To reach this conclusion, we considered the similarities across all parties, not just the 'CDU & CDU' as presented here.

Applying sentiment analysis to the manifestos produced the results shown in Figure 2. The chart illustrates the relative proportions of positive and negative sentiment sentences in each manifesto. The size of the dots corresponds to the total number of sentences in each manifesto. It can be observed that the 'AFD' has a relatively higher number of sentences with negative sentiment, but also a significant number of positive sentences. Conversely, the

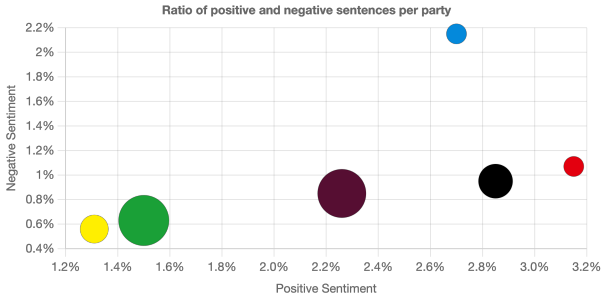

Figure 2. Sentimental expressions of parties

'SPD' has the highest relative number of sentences with positive sentiment. FDP' and 'B90/Green' seem to use more neutral language than the other parties. Furthermore, the figure shows that parties generally use more positive than negative language in their manifestos. Nevertheless, the figures are all relatively low. This is mainly due to the fact that manifestos tend to be written in a neutral tone, with parties avoiding overly emotional or strongly positive or negative language. Nevertheless, a trend can be observed, particularly with the 'AFD', which uses more negative language in its manifesto than other parties.

5 Conclusion

In conclusion, we utilised modern computational methods to analyse election manifestos and integrated advanced NLP techniques into a web application. We employed various analyses, including sentiment analysis at sentence level, similarity analysis using Cosine similarity, and various representation schemas such as TF-IDF, Word2Vec, and Doc2Vec. For the topic modeling analysis, we utilised a spaCy model with label categorisation. Additionally, we implemented a feature to display the top 30 words, excluding stopwords, and created a WordCloud based on these words. The results can be visualised using chart.js in a React App.

References

[1] E.D. Liddy, *Natural language processing* (2001)

[2] P.D. Katre, *Nlp based text analytics and visualization of political speeches* (2019)

[3] A.A. Efat, A. Atiq, A.S. Abeed, A. Momin, M.G.R. Alam, *Empoliticon: Nlp and ml based approach for context and emotion classification of political speeches from transcripts* (2023)

[4] S. Orellana, H. Bisgin, *Using natural language processing to analyze political party manifestos from new zealand* (2023)

[5] M. Pinta, P. Medina-Pérez, D. Riofrío, N. Pérez, D. Benítez, R.F. Moyano, *Automatic manifesto comparison using nlp techniques and the manifesto project domains - case study: 2021 ecuadorian presidential elections* (2021)

[6] M. Fenniak, *Extract text from a pdf*, https://pypdf.readthedocs.io/en/stable/user/extract-text.html#ocr-vs-text-extraction

[7] D. Lin, *An information-theoretic definition of similarity*, Presentation (1998), presented at the International Conference on Machine Learning, Madison, WI, USA, 24–27 July

[8] S. Vijayarani, M.J. Ilamathi, M. Nithya et al., *Preprocessing techniques for text mining-an overview* (2015)

[9] V. Balakrishnan, E. Lloyd-Yemoh, *Stemming and lemmatization: A comparison of retrieval performances* (2014)

[10] W.H. Gomaa, A.A. Fahmy et al., *A survey of text similarity approaches* (2013)

[11] D. Rani, R. Kumar, N. Chauhan, *Study and Comparision of Vectorization Techniques Used in Text Classification*, in *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (2022), pp. 1–6

[12] L. Mathew, V. Bindu, *A review of natural language processing techniques for sentiment analysis using pre-trained models*, in *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)* (IEEE, 2020), pp. 340–345

[13] B. Pang, L. Lee et al., *Opinion mining and sentiment analysis* (2008)

[14] B. Liu, *Sentiment analysis and opinion mining* (2022)

[15] M. Wankhade, A.C.S. Rao, C. Kulkarni, *A survey on sentiment analysis methods, applications, and challenges* (2022)

[16] O. Guhr, A.K. Schumann, F. Bahrmann, H.J. Böhme, *Training a broad-coverage German sentiment classification model for dialog systems*, in *Proceedings of the Twelfth Language Resources and Evaluation Conference* (2020), pp. 1627–1632

[17] Y. Goldberg, *A primer on neural network models for natural language processing.* (2016)

[18] R. Kohavi, *A study of cross-validation and bootstrap for accuracy estimation and model selection.* (1995)

[19] I. Goodfellow, Y. Bengio, A. Courville, *Deep learning* (2016)