Delft University of Technology, Bachelor Seminar of Computer Science and Engineering

Political Stance Detection using Knowledge Graphs and Sentiment Analysis

Abel Van Steenweghen , Pradeep Murukannaiah
TU Delft

Abstract

Sentiment analysis techniques estimate the opinion of the author of a text towards an entity from that text. Current sentiment analysis techniques are based on language features or deep learning methods. However, they do not make use of the extensive background knowledge that human readers can have. This makes it difficult for these models to detect irony, pop culture references and other subtleties for which connections between entities need to be known. The usage of knowledge graphs allows these models to use the enormous existing knowledge bases. We propose a political stance detection pipeline that makes use of knowledge graphs and sentiment analysis. The proposed pipeline uses a combination of existing deep learning methods and classic rule-based methods to train an opinion-aware knowledge graph, with which it classifies sentences as either liberal, conservative or neutral. The pipeline acts as both a classifier and a framework that can integrate existing stance detection models. In an experimental evaluation on the IBC and SemEval datasets, the proposed pipeline achieves an average F-score of 0.63, outperforming traditional machine learning models.

1 Introduction

We study the usage of knowledge graphs and sentiment analysis in the field of political stance detection. Feldman [8] defines sentiment analysis as "the task of finding the opinions of authors about specific entities." Sentiment analysis has many useful applications such as extracting information from a large set of product reviews, monitoring the reputation of a specific brand on social media, predicting the behavior of the financial markets and estimating the stance of voters towards certain issues during political campaigns [8].

We propose a model to estimate the political stance of a sentence or document, meaning a statement can be labelled as conservative, liberal or neutral. This can be used for estimating the overall ideology of an author. Estimating the ideology of an author is used in, for example, analyzing public opinion on social media. It can also be valuable because it allows you to look at an author's works more objectively, knowing their potential biases.

Current sentiment analysis algorithms focus on the overall sentiment or on a specified target entity. While the latter is already more specific it still only returns limited information about the text or its author. It cannot specify the stance towards entities outside of the text. For example, if someone writes: "Barack Obama was a great president." The current sentiment analysis algorithms would classify it as positive, and if they were target-specific it would select "Barack Obama" as target. However, more information can be extracted in this example. A human reader with background knowledge would be able to classify the stance of the author towards, for example, "The Democratic Party" to be also probably positive since there is a connection between the two entities. Knowledge graphs can provide the context and background knowledge necessary to make such connections [7].

This paper aims to answer the question:
How can knowledge graphs be applied in the field of political stance detection and can they improve the F-score?
The main focus of this research therefore lies on the development and testing of a political stance detection pipeline.

The proposed pipeline integrates existing models, both traditional and deep learning methods, with a knowledge base, as illustrated in Figure 1. This architecture allows the pipeline to be integrated with different models, allowing for application in other fields or for improved performance. The pipeline transforms unstructured text into a knowledge graph which is then enriched by linking it to an existing knowledge base (DBPedia[1]). Next, the opinions are linked towards the entities with a custom trained deep model. Finally, the stance towards a chosen entity is estimated. The advantage of using rule-based methods in the final stages is that the classification is more explainable than an abstract "black box" approach such as a deep learning approach.

The paper is structured as follows. First, the literature related to this work is discussed. Next, in Section 3, the design of the algorithm is discussed, with a focus on the most important stages of the pipeline. Sections 4 discusses the experimental setup and its results. Section 5 covers the responsibility of the research. Section 6 concludes the paper with the main takeaways and possible further improvements.

[1] https://www.dbpedia.org/

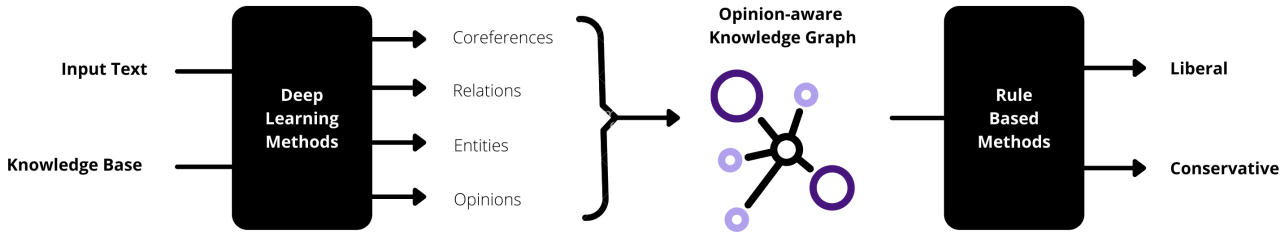

Figure 1: A pipeline for stance detection, combining deep learning and rule-based methods, and a knowledge base.

2 Related Work

Our work is related to two main research areas: sentiment analysis and knowledge graphs. This section presents the relevant existing work in both fields and discusses how it was used.

2.1 Sentiment Analysis

There exists a diverse set of sentiment analysis algorithms that fall in two categories: traditional models and deep learning models.

Examples of traditional models are Logistic Regression (LR) and Support Vector Machine (SVM) [14]. These models often make use of word embeddings to represent the sentence or document as a vector, which then can be incorporated by the classical machine learning techniques. Two examples of word embeddings are word2vec [10], where two vectors that are contextually close are also close in the vector space, and bag-of-words, which is a simple model that keeps track of the number of occurrences of each word in the text. A variation of this embedding is an n-gram, which acts the same as bag-of-words but counts the occurrences of n-tuples of sequential words. For example "I like this idea." contains the following tuples for a 2-gram: {"I like", "like this", "this idea"}.

The Sentiment-Specific Word Embedding model from [12] is another traditional model that uses sentiment lexicons in its word embedding. This means it uses opinion words and other sentiment-specific language to calculate similarity values and classify the sentences.

Examples of deep learning models are a Convolutional Neural Network (CNN) [16] and a Convolutional Bi-directional Long Short-Term Memory Network (CB-LSTM) [13]. Both [16] and [13] are applied in the context of political ideology detection. [16] ranked 1st and 2nd on task B and task A of the SemEval 2016 challenge [14] respectively. The CB-LSTM model of [13] aims to capture the context in which each target entity is expressed, by combining an LSTM with a CNN, which performs well at target-context representations.

Because of the good performance of these works a CNN architecture was chosen for the sentiment analysis part in the opinion linking stage at Subsection 3.5.

2.2 Knowledge Graphs

A knowledge graph is a network of entities connected by relations. It is often stored as set of triplets in the form of $[h, r, t]$ with h as head entity, r as relation, and t as tail entity. Relations can be directed or undirected, depending on its meaning. Knowledge graphs are not frequently used in the stance detection field despite advantages such as the creation of context-awareness, by combining meta-data and general text-mining approaches [5].

Our research mainly builds upon the works of Chen et al. (2017) [2] and of Xu et al. (2019) [18]. These works propose stance detection algorithms making use of opinion-aware knowledge graphs (OKG), with the application of political ideology estimation. While both these works focus on the importance of including background knowledge in the classification process they use different approaches.

Chen et al. [2] use a more traditional, rule-based approach working with the opinion-values linked to the entities and using other text features such as distance between opinion words and entities to estimate the sentiment.

Xu et al. [18] use a deep learning approach working with a knowledge graph embedding (KGE) to use the knowledge graph in a deep classifier. There has been a lot of research in the field of KGEs. A KGE "embeds the entities and relationships of multi-relational data in low-dimensional vector spaces" [1]. This way the information that a knowledge graph represents can be used in machine learning approaches. There are multiple methodologies for KGEs such as translation [1; 2], multiplication and neural networks based approaches.

This paper proposes a pipeline that builds on the OKG construction techniques from [2; 18] but introduce deep learning methods to improve certain stages.

3 Algorithm Design

The political stance detection pipeline consists of 7 phases:

1. Coreference Resolution
2. Entity Extraction and Linking
3. Relation Extraction
4. Knowledge Graph Formation

5. Opinion Linking
6. Opinion Propagation
7. Sentence Classification

3.1 Coreference Resolution

This first stage is a vital preprocessing step. A coreference is a word that refers to an earlier mentioned entity, called the referent. For example:

"Barack Obama is an American politician. **He** served as the 44th president of the United States."

⇓

"Barack Obama is an American politician. **Barack Obama** served as the 44th president of the United States." (A)

In natural language coreferences offer the advantage of avoiding repetitiveness. In the pipeline this is however disadvantageous for recognizing the entities in each individual sentence. By first resolving all coreferences to their referent, the algorithm can handle each sentence individually without missing certain entities. This process is implemented with the *neuralcoref* [2] package from *Huggingface*.

3.2 Entity Extraction and Linking

The second stage is entity extraction and linking. A named entity is defined as a "real world object" such as an organization, a person or a concept. Here the goal is to extract the entities out of each sentence and link them to an existing knowledge base. The knowledge base used is DBPedia[3]. Other candidates for entity linking were the WikiData [6] and the BLINK model [17] that both use Wikipedia as a knowledge base to retrieve the entities. DBPedia was selected because of its existing focus on knowledge graphs.

The entity linking makes use of the DBPedia Spotlight [4] tool [3]. There are two main hyperparameters that can be configured: confidence and support.

Confidence is a score for the disambiguation, meaning that a higher confidence score leads to more certainty that the linked entity is the right one and a lower confidence score leads to more (but possibly wrongly) recognized entities. Table 1 shows an example of this behavior. Note that at 0.3 confidence there are more entities recognized, but the "served" entity was linked to "serving in the Vietnam war", which is obviously wrong in this context.

Support is a score for the prominence, the number of inlinks in Wikipedia, of the entities. A high support means an entity is more prominent and more probable to be the right one. A low support leads again to more recognized entities. For our implementation the support hyperparameter was set at 10, which is relatively low, thereby allowing less prominent entities to be recognized when they pass the confidence score.

Table 1: Influence of the confidence parameter on the number of recognized entities in sentence A in Section 3.1

Confidence	Recognized Entities
0.9	Barack Obama
0.5	Barack Obama, American, United States
0.3	Barack Obama, American, Politician, Served, President, United States

3.3 Relation Extraction

The next stage is relation extraction. Here each sentence is individually analyzed by going over each permutation of two entities of the sentence. This is done by using the open source project *OpenNRE*[5] [9]. *OpenNRE* makes use of CNN and BERT encoders to gain a state-of-the-art relation extraction. The model takes each permutation of two entities as input and predicts whether there is a relation between them and what that relation might be. For our pipeline the default pre-trained model was used, but OpenNRE allows for training an own model, which could further increase performance.

3.4 KG Formation

This stage forms a knowledge graph from the extracted entities and relations. The entities act as the nodes in the knowledge graph, and the relations as the edges connecting the entities. A KG can be described by the Resource Description Framework (RDF), which is used to represent linked data. In RDF, a KG is described as a set of triples. An RDF triple contains a subject, a predicate and an object. The predicate represents the relation between two entities, extracted in section 3.3. Figure 2 shows the format of an RDF triple in graph form.

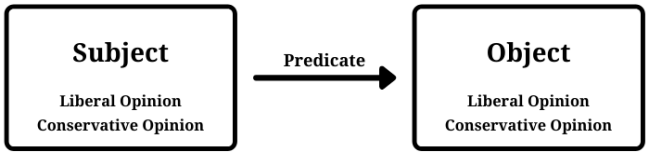

Figure 2: An RDF Triple.

In this work, we use a modified version of an RDF triple where both nodes, the subject and the object, also contains the liberal and conservative opinions towards the specific entity. Each opinion is a value between -1 and 1. This is described in detail in the next section.

Neo4J [6] was used as database for the implementation because of its graph-centric approach and great performance [4].

[2] https://github.com/huggingface/neuralcoref
[3] https://www.dbpedia.org/
[4] https://github.com/dbpedia-spotlight/dbpedia-spotlight
[5] https://github.com/thunlp/OpenNRE
[6] https://neo4j.com/

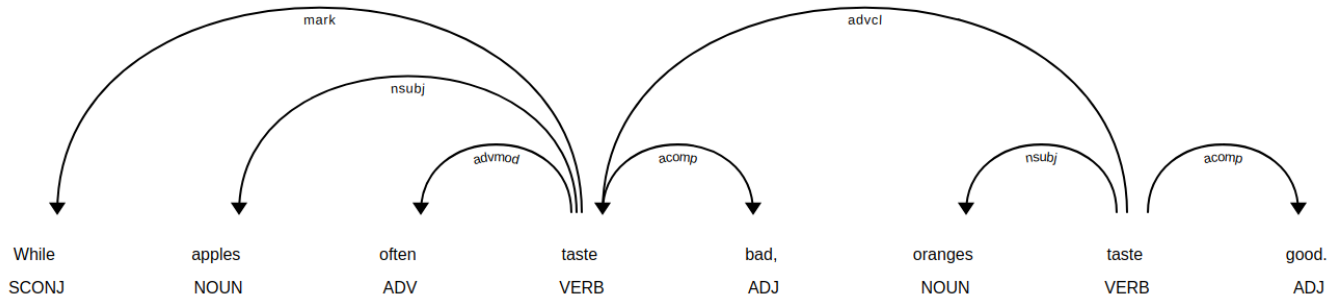

Figure 3: POS tags and dependencies of sentence B, visualized with displaCy[7].

3.5 Opinion Linking

This stage estimates the sentiments expressed in the sentence towards each occurrence of an entity and aggregates them into a liberal and conservative opinion. For this stage existing target-specific sentiment analysis models can be used, the only requirement is that the sentiment scores are aggregated and normalized to be in the $[-1, 1]$ interval, where -1 means absolutely opposed and 1 means absolutely in favor. The implementation of this paper is a variation on the work of Chen et al. [13].

The model of [13] makes use of the distance between the opinion words and the entity to link the entity with a sentiment score. An opinion towards an entity v is calculated as a vector $[libOp, conOp]$ determined by the aggregation of all the separate occurrences, following these aggregation functions:

$$\begin{cases} libOp = \sum_{L_i \in L} \sum_{op_j \in L_i} \frac{op_j \cdot SO}{d(op_j, v)} \\ conOp = \sum_{C_i \in C} \sum_{op_j \in C_i} \frac{op_j \cdot SO}{d(op_j, v)} \end{cases} \quad (1)$$

L is the set of liberal training sentences and C is the set of conservative training sentences. op_j is an opinion word in a sentence L_i or C_i. For recognizing the opinion words an opinion lexicon is used, this is a dictionary that stores a set of known opinion words with their respective sentiment orientations. $d(op_j, v)$ is the distance between the current estimated entity v and the opinion word op_j. The distance between two words is defined as the minimum number of words between the first word and the second word, in this case the entity word and the opinion word. $op_j.SO = \{-1, 0, 1\}$ is the notation for the sentiment orientation of the opinion word, meaning $\{$negative, neutral, positive$\}$ respectively.

The approach of Equation 1 of using opinion words and their distance towards the entities has two main disadvantages. First, there can be multiple entities in one sentence with conflicting opinion words. For example, the sentence:

"While apples often taste bad, oranges taste good." (B)

has two entities, "apples" and "oranges", and two conflicting opinion words, "good" and "bad". If we would follow Equation 1 and use distance to compute the opinion towards "oranges", "bad" (distance of 1) would be more related to "oranges" than "good" (distance of 2), which is clearly false. A second disadvantage is that the opinion lexicon, the dictionary of opinion words, can be limiting. There may be parts of the sentence that do have a sentiment but are not recognized.

Therefore a variation of formula 1 is implemented:

$$\begin{cases} libOp = \frac{1}{|L|} \sum_{L_i \in L} \left(\frac{1}{|L_i|} \sum_{d_j \in L_i} s(d_j) \right) \\ conOp = \frac{1}{|C|} \sum_{C_i \in C} \left(\frac{1}{|C_i|} \sum_{d_j \in C_i} s(d_j) \right) \end{cases} \quad (2)$$

d_j is a dependency towards the entity L_i or C_i. $|L_i|$ and $|C_i|$ are the numbers of dependencies in the entity L_i and C_i respectively. They are used to normalize the sum of aggregated sentiments. $|L|$ and $|C|$ are the numbers of recognized entities in the sentence L and C respectively. They are used to normalize the sum of aggregated entities.

A dependency is a part of a sentence that depends on another part and thereby indicates a connection. To resolve these dependencies the algorithm uses Part-of-speech (POS) tags. Figure 3 shows the POS tags of sentence B. Here "oranges" gets the dependency "taste good". The resulting dependencies are based on the semantic structure of the sentence, which is more reliable than a simple distance measure. In our implementation the dependencies are extracted with the DependencyMatcher[8] from Spacy.

$s(d_j)$ is the estimated sentiment of the part d_j. Our implementation uses TextBlob[9], which returns the sentiment polarity and assessments. TextBlob uses a pretrained model, following a CNN architecture, as in Section 3.3 this model could be optimized to further increase performance.

3.6 Opinion Propagation

After the opinion linking stage the model can already use the opinion-aware knowledge graph (OKG) in the final stage of sentence classification. But to improve the accuracy the model can leverage the relations between the entities to propagate the opinions of known entities towards the unknown

[8] https://spacy.io/api/dependencymatcher
[9] https://github.com/sloria/TextBlob

entities. This is done with the following formula:

$$\begin{cases} libOp = -\sum_{j=1}^{p} \log\left[P\left(r_j\right)\right] \cdot libOp_j \\ conOp = -\sum_{j=1}^{p} \log\left[P\left(r_j\right)\right] \cdot conOp_j \end{cases} \quad (3)$$

This formula iterates over each entity j connected by r_j to the current unlabeled entity. $P(r_j)$ is the probability of the relation r_j in the OKG. The probability is calculated as follows:

$$P(r_j) = \frac{|r_j|}{|R|} \quad (4)$$

$|r_j|$ is the number of occurrences of the type of relation r_j, such as, for example, "MEMBER_OF". $|R|$ is the number of all relations in the OKG. The assumption here is that specificity is important to relevance. So a relation that appears less in the OKG is given more relevance than a frequent occurring relation. This idea is used in both [13] and [15].

3.7 Sentence Classification

Going into this stage the model has been trained and has constructed an OKG G, storing an individual opinion vector for each entity. When the model is applied on a new sentence or document T from the test set, the first step is to get the intersection of the set of entities in G and the set of entities in T. We call this intersection V.

The second step is to get the opinion orientation OO_v towards each entity v of V. It follows this formula:

$$OO_v = \text{sign}\left(\sum_{d_v \in T} s(d_v)\right) \quad (5)$$

As in formula 2, d_v are the POS dependencies linked to the handled entity v and $s(d_v)$ is the estimated sentiment. The sentiments are aggregated and a sign function is applied to get a value of -1, 0 or 1, meaning again {Favor, Neutral, Opposed}.

After the opinion orientation of the entity is calculated it is compared to the opinions of the entities from the OKG in V.

$$ideology = \frac{\sum_{v \in V}(libOp_v \cdot OO_v - conOp_v \cdot OO_v)}{|V_T|} \quad (6)$$

Here V_T is the set of entities in the sentence T and $|V_T|$ is the number of entities in in V_T. The overall ideology of sentence T will be labeled as liberal if $ideology > n$ then , as conservative if $ideology < -n$, and as neutral if $-n \leq ideology \leq n$. Here n is a threshold hyperparameter that sets an interval $[-n, n]$ between which the ideology is not distinctively liberal or conservative enough, wherein therefore samples get labeled as neutral. In our implementation this n is 0.1.

The following example illustrates the classification process.

"**President Trump**'s horrifying **energy policy**, which focuses on the reduction of **American petroleum** regulation, will cause a significant raise in damaging **greenhouse-gasses**." (C)

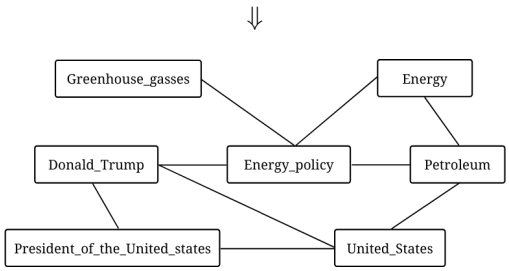

Figure 4: A subgraph of the intersecting entities of the text C and the OKG G. The edges are the relations between the entities that are present in the text. Other relations connected to the entities are omitted for clarity.

Table 2: The opinion orientation and opinion scores of the recognized entities from text in C. Because of the overall negative sentiment in C all the opinion orientations are negative but they may differ.

Entity v	OO_v	$LibOp_v$	$ConOp_v$
Greenhouse_gasses	-1	-0.53	-0.21
Energy	-1	0.32	0.43
Donald_Trump	-1	0.78	-0.63
Energy_policy	-1	0.45	- 0.28
Petroleum	-1	0.38	-0.54
POTUS	-1	0.34	0.23
United_States	-1	0.54	0.43

$$ideology = 0.29 = \textbf{Liberal} \quad (7)$$

4 Evaluation

This section evaluates the performance of the algorithm. We measured performance with the F-score and compared it against three other traditional models on two datasets.

4.1 Datasets

The algorithm was tested on two datasets, both in the political context.

The SemEval2016 dataset [14] consists of tweets during the US 2016 presidential election. It covers six subjects: Hillary Clinton (HC), Donald Trump (DT), the

feminist movement (FM), the Legalization of Abortion (LA), Atheism (AT) and climate change is a Real Concern (CC). Each tweet is annotated with a target (one of the subjects) and a stance (in favor, opposed, neutral). For our model the data is preprocessed by labeling each tweet as either liberal (Lib), conservative (Con) or neutral. Table 3 describes how the known stance of the ideologies towards the subjects are applied to achieve this.

Table 3: Political ideology per subject for the SemEval2016 dataset.

Subject	Favor	Opposed	Neutral
HC	Lib	Con	Neutral
DT	Con	Lib	Neutral
LA	Lib	Con	Neutral
CC	Lib	Con	Neutral
AT	Con	Lib	Neutral
FM	Lib	Con	Neutral

The IBC dataset [11] consists of sentences extracted from works of the Ideological Books Corpus, which is a collection of books and articles, written between 2008 and 2012 by authors of whom the political leaning is well known. Each sentence is annotated as either liberal, conservative or neutral.

Tables 4 and 5 describe the distributions of the datasets.Both datasets have an already established train/test distribution, therefore the train/test ratio differs.

Table 4: Train/Test distribution of SemEval and IBC datasets.

Dataset	#Total	#Train	#Test	Train/Test
SemEval	4.9K	3K	1.9K	61/39
IBC	4.3K	3.4K	0.9K	79/21
Combined	9.2K	6.4K	2.8K	70/30

Table 5: Ideology distribution of SemEval and IBC datasets.

Dataset	#Total	#Lib	#Cons	#Neutral
SemEval	4.9K	1.7K	1.9K	1.3K
IBC	4.3K	2K	1.7K	0.6K
Combined	9.2K	3.7K.	3.6K	1.9K

4.2 Models

The pipeline was compared against three types of models.

- LR: Logistic Regression algorithm using bag-of-words feature embedding.
- SVM: Support vector machine algorithm using bag-of-words feature embedding.
- CNN: A convolutional neural network algorithm using word2vec [10] embeddings.

The LR and SVM model were both implemented with scikitlearn[10]. The CNN model was implemented with the TextCategorizer[11] component from Spacy.

4.3 Performance Measure

The performance of the models is measured with the F-score. This performance measure combines two concepts, precision and recall, to evaluate the classifications. The formulas use three types of classification:

- True Positive (TP): Correctly classified as the specific ideology.
- False Positive (FP): Wrongly classified as the specific ideology.
- False Negative (FN): Wrongly classified as another ideology than the specific ideology.

Precision (P) and recall (R) are defined as follows:

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$F = \frac{2 * P * R}{P + R}$$

Since the F-score is a measure that only works for binary classifications (right or wrong classification) the F-score is calculated for each ideology separately. Thereafter the average of the three ideologies is calculated:

$$F_{\text{Ideology}} = \frac{2 * P_{\text{Ideology}} * R_{\text{Ideology}}}{P_{\text{Ideology}} + R_{\text{Ideology}}}$$

$$F = \frac{F_{\text{Liberal}} + F_{\text{Conservative}} + F_{\text{Neutral}}}{3}$$

4.4 Experimental Results and Analysis

Tables 6 and 7 contain the F-scores of the models of section 4.2 and the knowledge graph pipeline (KGP) as described by section 3. The best F-scores per ideology are marked in bold.

The knowledge graph pipeline outperformed both the 2 classical machine learning models and the deep learning model on the overall F-score. The second-best performing model is the CNN, this is in line with the good performances of [13; 16], discussed in Subsection 2.1. CNN outperforms KGP on both the liberal set from the SemEval dataset and the conservative set from the IBC dataset. These two sets are both

[10] http://scikit-learn.org/stable/index.html

[11] https://spacy.io/api/textcategorizer

Table 6: F-scores of different models on the SemEval dataset.

Model	F	F_{lib}	F_{cons}	$F_{neutral}$
LR	0.5564	0.5632	0.5328	0.5733
SVM	0.5795	0.5832	0.5623	0.5929
CNN	0.6120	**0.6401**	0.5935	0.6024
KGP	**0.6276**	0.6279	**0.6203**	**0.6346**

Table 7: F-scores of different models on the IBC dataset.

Model	F	F_{libe}	F_{cons}	$F_{neutral}$
LR	0.5925	0.5871	0.5943	0.5957
SVM	0.6117	0.6245	0.6002	0.6103
CNN	0.6296	0.6237	**0.6329**	0.6323
KGP	**0.6356**	**0.6381**	0.6166	**0.6521**

the smaller sets in their respective dataset. This may indicate that CNN is more efficient than KGP, allowing it to get better performance with a smaller dataset.

Noteworthy is that every model performed better on the IBC dataset than on the SemEval dataset, this may indicate a correlation between the size of the training set and the performance for all models.

D and E are examples of a liberal and a conservative sentence respectively, that were classified wrongly as neutral by the baseline models, but correctly by the KGP model. The recognized entities are marked in bold.

> "'We share **culture**, we share friends and enemies', explains **Mara Keisling**, a longtime **trans activist** who for the past five years has led the National Center for **Transgender** Equality." (D)

> "Indeed, **Eric Garris** wrote a generally favorable piece about Brown for Reason in 1975, and **Murray Rothbard** praised him that same year in **The Libertarian Forum**, though his later remarks about the **governor** were more **caustic**." (E)

Remarkable about D is that there are no clear opinion words or explicitly stated sentiments. This may indicate that KGP has a lower reliance on clearly stated sentiments and can rely more on the sentiments that are stored in its OKG.

Remarkable about E is the presence of two persons and one organization as recognized named entities. After closer investigation of the OKG, these two persons had both a conservative leaning sentiment and had only a combined total of 8 relations towards other entities. This may indicate that KGP is better at classifying sentences with multiple less known entities.

5 Responsible Research

This paper is written in the context of the CSE3000 "Research Project" course from Delft University of Technology. The field of political ideology detection and the field of politics in general is ethically sensitive because of its prominent influence over multiple aspects of a society. Therefore it is important that there is no bias in the classifications and that this work is reproducible.

5.1 Bias

The classifications of the algorithm are susceptible to bias from three main sources:

- Third-party models, used for coreference, entity, relation and opinion extraction. Since these models are partly out of the control of the pipeline this may skew the classifications towards a particular ideology.
- Knowledge base, used for linking entities. Some entities may be not represented in the knowledge base.
- Datasets, used to train and test the pipeline. If one ideology has a considerable majority in the data, the results may be biased towards this ideology. The SemEval dataset was slightly more liberal and the IBC dataset was slightly more conservative, making the combined dataset equally distributed between both ideologies, see table 5.

5.2 Reproducibility

To make reproducibility manageable, the stages of the pipeline and the used packages have been described in detail in section 3. The pipeline was implemented with the Spacy 3.0 framework and all used packages have been annotated with footnotes. The code is also accessible at GitHub[12]. The structure and preprocessing of the datasets have been described in section 4.1. There is no significant influence of randomness throughout the whole pipeline.

6 Conclusion

Knowledge graphs and sentiment analysis techniques can be combined to create a political stance detection algorithm. The knowledge graph pipeline (KGP) proposed in this paper offers a framework for existing stance detection models to use the background knowledge stored in knowledge bases and improve their context-dependent classifications.

The results showed that KGP achieved an average F-score of 0.63 outperforming three other traditional models on the IBC and SemEval dataset. Closer investigation of the classifications showed that KGP is superior in classifying sentences with no explicitly stated sentiments or with sentences with multiple named entities.

[12]https://github.com/Abel-VS/Grapher

6.1 Future Work

There is a lot of room for improvement and further features for this pipeline. First, there is the subject of explainability. Most deep learning methods act like a "black box", where a model achieves high efficiency and performance but the classifications can't be given an explanation. By combining the efficiency of modern deep learning methods such as CNN and BERT with the clarity of the rule-based opinion propagation and sentence classification, the developed model should be able to offer intuitive explanations for its classifications. This can be valuable in fields such as medical research where a decision needs to be backed by a sound explanation.

A second improvement could be to integrate the links from the KB entities in the OKG. Currently the relations between the entities are purely extracted from the training data itself, but the structure of DBPedia already contains many links between its entities. If this were to be implemented the pipeline could recognize relations to entities that were not mentioned in the training data.

7 Acknowledgments

I would like to thank Pradeep Murukannaiah for his guidance as supervisor and responsible professor through the whole research and writing process. I am also grateful for the help and feedback of Simon Marien, Wout Haakman, Kristof Vass, and Jacob Roeters, with whom I formed a peer group.

References

[1] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[2] W. Chen, X. Zhang, T. Wang, B. Yang, and Y. Li. Opinion-aware knowledge graph for political ideology detection. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[3] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, 2013.

[4] D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J. L. Larriba-Pey. Survey of graph database performance on the hpc scalable graph analysis benchmark. *Web-Age Information Management Lecture Notes in Computer Science*, page 37–48, 2010.

[5] J. Dörpinghaus and A. Stefan. Knowledge extraction and applications utilizing context data in knowledge graphs. *Proceedings of the 2019 Federated Conference on Computer Science and Information Systems*, 2019.

[6] M. Farda-Sarbas and C. Müller-Birn. Wikidata from a research perspective - a systematic mapping study of wikidata, 2019.

[7] B. Fatemi, S. Ravanbakhsh, and D. Poole. Improved knowledge graph embedding using background taxonomic information. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3526–3533, 2019.

[8] R. Feldman. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89, 2013.

[9] X. Han, T. Gao, Y. Yao, D. Ye, Z. Liu, and M. Sun. OpenNRE: An open and extensible toolkit for neural relation extraction. In *Proceedings of EMNLP-IJCNLP: System Demonstrations*, pages 169–174, 2019.

[10] T. M. G. Inc., T. Mikolov, G. Inc., I. S. G. Inc., I. Sutskever, K. C. G. Inc., K. Chen, G. C. G. Inc., G. Corrado, J. D. G. Inc., and et al. Distributed representations of words and phrases and their compositionality, Dec 2013.

[11] M. Iyyer, P. Enns, J. Boyd-Graber, and P. Resnik. Political ideology detection using recursive neural networks. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014.

[12] Q. Li, S. Shah, R. Fang, A. Nourbakhsh, and X. Liu. Tweet sentiment analysis by incorporating sentiment-specific word embedding and weighted text features. *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2016.

[13] X. Li, W. Chen, T. Wang, and W. Huang. Target-specific convolutional bi-directional lstm neural network for political ideology analysis. *Web and Big Data Lecture Notes in Computer Science*, page 64–72, 2017.

[14] S. Mohammad, S. Kiritchenko, P. Sobhani, X. Zhu, and C. Cherry. Semeval-2016 task 6: Detecting stance in tweets. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016.

[15] M. Schuhmacher and S. P. Ponzetto. Knowledge-based graph document modeling. *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014.

[16] W. Wei, X. Zhang, X. Liu, W. Chen, and T. Wang. pkudblab at semeval-2016 task 6 : A specific convolutional neural network system for effective stance detection. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016.

[17] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer. Zero-shot entity linking with dense entity retrieval, 2020.

[18] Z. Xu, Q. Li, W. Chen, Y. Cui, Z. Qiu, and T. Wang. Opinion-aware knowledge embedding for stance detection, 2019.