Supplementary Material for "Introducing an Interpretable Deep Learning Approach to Domain-Specific Dictionary Creation: A Use Case for Conflict Prediction"

Sonja Häffner[1], Martin Hofer[1], Maximilian Nagl[2], and Julian Walterskirchen[1]

[1]*Kompetenzzentrum Krisenfrüherkennung, CISS, Universität der Bundeswehr München Email: julian.walterskirchen@unibw.de*
[2]*Lehrstuhl für Statistik und Risikomanagement, Universität Regensburg*

Appendix

Software Implementation

With the exception of the two word scaling techniques (Wordscores and Wordfish) that were implemented in R with the Quanteda and Quanteda Textmodels packages, all other approaches have been implemented using Python 3.9. The neural networks for the dictionary creation were built utilizing Keras while the BERT model was implemented applying the Transformers library (Hugging Face) in conjunction with Pytorch. The computation of the feature importance scores was mostly performed with the Autograd Numpy library. Other libraries involved include Sklearn, especially for evaluation metrics and the creation of the document term matrix, as well as Nltk and Spacy for text preprocessing. More information on software requirements can be found at https://doi.org/10.7910/DVN/Y5INRM.

Sensitivity analysis

To extract the relative feature importance of our input words, this paper employs the feature importance metric defined by Horel *et al.* (2018) (sensitivity analysis). Technically, the sensitivity of a model is defined as the derivative of a model's output with respect to its inputs. Although it is possible to use a number of different feature importance techniques for the creation of our dictionary (see e.g. Ribeiro, Singh, and Guestrin 2016; Lipovetsky and Conklin 2001; Hooker *et al.* 2018; Samek *et al.* 2016), according to Horel *et al.* (2018), sensitivity analysis is an especially suitable approach for assessing the predictions of a neural network. First, assessing a model's output with respect to its input is a very intuitive way to explain the predictions of a neural network. Second, neural networks are naturally differentiable as the training process involves taking the derivative of the loss function with respect to the weights of a neural network. Sensitivity analysis is consequently computationally cheap as the libraries used to train the network can be leveraged in order to compute the derivatives. Horel *et al.* (2018) showed that their approach was able to produce similar results compared to the Local Interpretable Model-agnostic Explanations method (LIME, a popular feature importance metric that focuses on identifying the contribution a feature has on the target variable for specific observations) while being much faster to compute (0.5 seconds for sensitivity analysis versus 6 hours for LIME). Third, sensitivity analysis offers two types of model explanations as the derivatives can be aggregated locally or globally. Finally, sensitivity analysis can be applied to many different neural network architectures, such as fully connected Neural Networks (NNs), Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs).

Political Analysis (2023)

DOI: 10.1017/pan.xxxx.xx

Corresponding author
Julian Walterskirchen

Edited by
John Doe

© The Author(s) 2023. Published by Cambridge University Press on behalf of the Society for Political Methodology.

Preprocessing steps

All of the CrisisWatch documents were carefully preprocessed according to standard NLP practices. It is important to mention that the preprocessing steps for the BERT Models diverge from the ones for the dictionary creation. This is due to the fact that BERT models are directly trained on raw texts and bidirectionally work with sequences. Bidirectional means that the left and the right context of a sequence/sentence is taken into account. Therefore, the exclusion of stopwords (e.g. the, are, and, …), lemmatization as well as the removal of punctuation is strongly discouraged as they change the context of a sentence.

For the creation of the dictionary, the following standard preprocessing steps have been applied. First, the CrisisWatch reports have been scraped from the ICG Website, and the remaining HTML characters have been removed. In addition, texts have been lowercased and special characters (such as punctuation and numbers) have been removed. After tokenizing the texts, the words were lemmatized and standard stop words were removed. For creating the document term matrix that serves as the input of the neural network, the most frequent 3,000 words were identified, additionally, the top 1000 bi-grams were added to that list. In addition, we manually removed words related to organizations (international organizations, companies, armed groups, …), people (their name) or location (such as city and country names) to ensure that the dictionary remains general enough. Although for other settings this step might be optional, for us it was crucial that the neural network does not learn that a certain country or certain people are associated with conflict. In contrast, we intend the neural network to pick up other trends and nuances associated with conflicts, such as the occurrence of protests or the absence of reports on military clashes.

As most algorithms can only work with numerical inputs, the text data has to be transformed into a structured numerical format. The use of document term matrices is one of the most popular methods (Khadjeh Nassirtoussi *et al.* 2014) that transforms documents into word vectors, thereby assuming independence among words. In a document term matrix, each row corresponds to a single document while each column corresponds to one word of the overall vocabulary of all documents. As this leads to an exploding feature space, it is common practice to reduce the vocabulary size drastically (Pestov 2013).

Instead of using simple word counts, term frequency-inverse document frequency (tf-idf) scores were calculated in order to weigh the identified words in the document term matrix. Consequently, each entry in the document term matrix corresponds to the tf-idf score of a word t in document d. In order to obtain the tf-idf score of a word, its term frequency (how often does word t appear in document d) is multiplied by its inverse document frequency (in how many documents does word t appear). The idea behind tf-idf scoring is to give a higher score to words that only appear in a few documents (supposedly more important words) compared to words that appear in all documents (supposedly less important words).

We also want to discuss why we decided to work with a document term matrix instead of employing a Long Short-Term Memory (LSTM) model that makes use of word dependencies. The main rational of our approach is to build a model that is complex enough to learn possibly non-linear patterns in our data while being computationally inexpensive. LSTMs work sequentially with data (therefore making it difficult to speed up computation time)and struggle, when the sequences get too long. Therefore, we decided to work with a "simple" deep feed forward neural network as it is much more efficient with regards to computation time and should in theory, according to the universal approximation theorem be able to approximate any arbitrarily complex function (see Hornik, Stinchcombe, and White (1989)). As well, when we look at our results, we can see that our approach outperforms a BERT model that makes use of those word dependencies. We therefore conclude that the performance of our "simple" neural network is sufficiently good (even better than models that make use of word dependencies) and do not think that the application of LSTMs will boost performance considerably (if at all).

Comparison Models

As the performance of our dictionary is compared to the performance of general-purpose dictionaries, the sentiment scores are calculated for the Vader and HGI4 Dictionary. For the two sentiment dictionaries, this score represents how many words with a "positive" or "negative" connotation are mentioned in the respective document. These scores can be understood as representing the overall tonality of each report and should not be interpreted as a document's stance toward a specific topic.[1]. Both Vader and HGI4 have found broad applications in various fields and are generally considered general-purpose sentiment dictionaries. HGI4 consists of classifications of over 3,000 word strings according to their negative and positive connotation.[2] Vader, similarly, is a general-purpose sentiment dictionary, that combines a weighted word list with "five general rules that embody grammatical and syntactical conventions for expressing and emphasizing sentiment intensity" (Hutto and Gilbert 2014, p.225).

The sentiment score of the HGI4 dictionary is calculated using the Python library Pysentiment2 (DeRobertis 2020). The score is calculated as follows:

$$Polarity = \frac{Pos - Neg}{Pos + Neg} \tag{1}$$

Note that Pos and Neg refer to simple word counts for positive and negative words. The sentiment score of the Vader dictionary is calculated using the Python library Nltk (Bird, Klein, and Loper 2009). It is the normalized sum of valence scores that are calculated applying certain rules, called heuristics.[3] The compound score then is simply: $compound = \frac{x}{x^2+\alpha}$, where α is a normalization constant that is set to 15. Finally, the different scores serve as the sole input data to several Random Forest and XGBoost Models predicting fatalities (regression task).

As PETRARCH2 is a system that uses dictionaries to extract conflict events, it does not provide a score that can be directly applied in a prediction task. Hence, for our CAMEO scale variable, we assigned each event a score based on a cooperation-conflict scale that has its origin in the article by Goldstein 1992 but has been adjusted to the CAMEO dictionary. The scale we used was produced by Philip Schrodt and can be found online at https://parusanalytics.com/eventdata/cameo.dir/CAMEO.SCALE.txt. The CAMEO score was calculated by assigning the score of each event category (we used the top level of categories) and aggregating them on the country-month level.

Document Scaling

We also apply two popular text-scaling techniques: Wordscores (Laver, Benoit, and Garry 2003) and Wordfish (Lo, Proksch, and Slapin 2016). The idea behind text-scaling approaches is similar to the idea behind dictionary approaches. Documents that use similar language (measured as word frequencies) should also represent similar concepts. While Wordscores belong to the supervised machine learning techniques and require reference texts as inputs, Wordfish belongs to the unsupervised machine learning techniques and infers positions in documents solely on the basis of word frequencies and how they co-occur across the corpus. We apply both techniques to our corpus of conflict reports. As the Wordfish approach is an unsupervised method, we directly apply it to the conflict reports and calculate document scores.[4] For the Wordscore approach, we first define a fraction of the respective documents in each corpus as reference texts and the correspond-

1. See Bestvater and Monroe (2022) for an extensive discussion what sentiment constitutes and how it differs from stance detection.
2. See e.g. Loughran and McDonald (2011) for a discussion of the Harvard IV-4 dictionary.
3. The reader is referred to the original paper (Hutto and Gilbert 2014) for further information on the valence scores.
4. Wordfish, therefore, is maybe at a disadvantage compared to Wordscore, as it is not explicitly trained on our outcome of interest. However, the underlying assumption that reports about conflict dynamics should use similar language depending on the severity of the conflict, should still hold, making the use of Wordfish as one of our benchmark models a sensible choice.

ing fatalities as reference scores. Those references are then used to predict the positions of the remaining texts. As before with the (sentiment) dictionary approach, we determine how closely those scores are aligned with fatalities and build simple models predicting fatalities.

Transformer Models

As currently transformer models are considered state-of-the-art models in the field of NLP, we fine-tune a Bidirectional Encoder Representations from Transformers (BERT) model to compare its performance to our dictionary approach. The success of BERT models is due to a few factors, including advancements in computing power as well as the transformer architecture that allowed for the training of massive corpora with over 3 Billion words. These models are trained in a way so they can retain information about each word's position and relationship to other words, allowing them to model language on a higher level of complexity. Additionally, BERT models leverage the attention algorithm that allows the model to only focus on important parts of a sentence. Further information and a more technical description of BERT models can be found in Devlin *et al.* (2018).

Originally, transformer models are designed to solve classification problems. However, as we build our dictionary on fatalities, it is necessary to rebuild the final layer of the transformer such that it can solve a regression task. As BERT Models are trained on whole sentences or documents, they do not require the same preprocessing steps as applied to the text for the dictionary creation. In our case, only HTML characters and numbers were removed. The minimally preprocessed texts serve as input data for the transformer model predicting fatalities. As before, the dataset is split into train, validation, and test set and the models are fine-tuned on a GPU cluster. Employing a GPU cluster was necessary as the fine-tuning of the transformer model quickly became unfeasible on regular computers (with 64 GB RAM), underlining the high computational costs of this approach. Due to the relatively small size of the dataset all layers, except the last one, were frozen, consequently, only the last layer was fine-tuned. This technique is frequently used in NLP tasks to avoid overfitting and speed up the training process and should not affect out-of-sample prediction results (Lee, Tang, and Lin 2019). It is common practice to choose transformer models that have been pre-trained on a similar task in order to obtain a better performance. Given that we are dealing with documents related to conflict, instead of the classical BERT model, we chose the ConfliBERT (Hu *et al.* 2022) model as it has been pre-trained on multiple conflict-related corpora. Similar to the BERT model, the architecture has 12 layers, 768 hidden units, 12 attention heads, and 110M parameters in total and was trained on over 7 billion words (34 GB) from conflict-related reports (organizations and governments) and news articles.

Results Lasso

The following table shows the results for the weighted feature importance scores for all different approaches including the scores obtained from creating a dictionary using Lasso regression.

Hyperparameters Evaluation Models

This section shows the hyperparameters optimized for the two evaluation models.

Unweighted Feature Importance Score

This section presents the results of the comparison of the unweighted feature importance score, implemented as a simple word count adjusted by document length. We calculated this score for our OCoDi, the HGI4 dictionary, and the Lasso model, unfortunately for the CAMEO and Vader dictionary, we were not able to obtain such scores. As an event extraction dictionary, PETRARCH extracts events from sentences and among other things, returns the event type. We used this event type and mapped it on a CAMEO scale measuring degrees of conflict and cooperation and used

Table 1. Results of predicting fatalities with different approaches including the Lasso regression model.

Model	OCoDi	Vader	HGI4	Wordfish	Wordscore	CAMEO	Lasso
Random Forest							
MSE	**1.59**	2.61	3,13	4,39	2,20	2.68	1.65
R^2	**0.64**	0.40	0.29	-0.00	0.50	0.39	0.62
XGBoost							
MSE	**1.60**	2.60	2.99	4.40	2.21	2.65	1.68
R^2	**0.63**	0.41	0.32	-0.00	0.50	0.39	0.62
	ConfliBERT						
MSE	1.75						
R^2	0.60						

Table 2. Hyperparameter constellations for Random Forest and XGBoost.

Model	OCoDi	Vader	HGI4	Wordfish	Wordscore	CAMEO	Lasso
Random Forest							
Trees	1100	1400	1300	600	1200	800	600
Depth	7	7	7	7	7	7	7
XGBoost							
Boosting Stages	400	400	100	100	100	100	800
LR	0.05	0.05	0.05	0.2	0.1	0.1	0.2
Depth	3	3	6	3	6	3	3
Min. Child	100	100	1	1	100	10	100

the scale directly to predict the natural logarithm of fatalities. For the Vader dictionary, the Nltk package does only return a compound score as well as probabilities for the respective document to be positive, negative, or neutral.

Table 3. Model results for the unweighted dictionary scores.

Model	OCoDi	HGI4	Lasso
Random Forest			
MSE	**1.62**	2.95	1.92
R^2	**0.63**	0.33	0.56
XGBoost			
MSE	**1.62**	2.98	1.98
R^2	**0.63**	0.32	0.55

Prediction Clusters

In Figure 1, we have arranged the data frame in a time-series cross-section format, so observations for the same country appear next to each other in the plot, we can see that a large part of our under- and over-estimations seem to be driven by a couple of countries.

Figure 1 also shows how many of the large differences between our predictions and the observed values are driven by clusters of countries that experienced very high numbers of monthly fatalities (signified by the high positive values). This should not be too discouraging, given that other variables

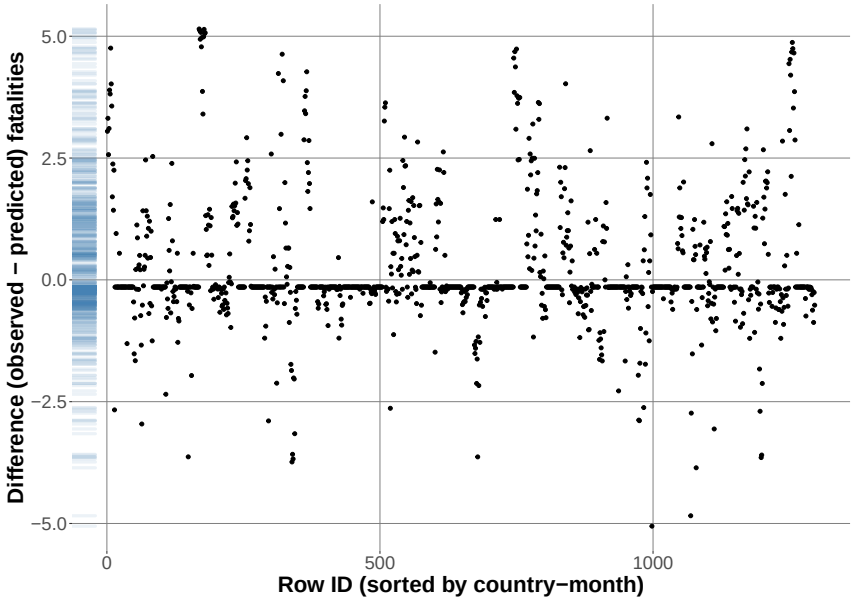

Figure 1. Observed vs. difference (clustered by country-month), Random Forest

normally used in conflict prediction models fare reasonably well at identifying ongoing, high-intensity conflicts.

Data Availability Statement

Replication code and data for this article are available at Häffner *et al.* (2023) at https://doi.org/10.7910/DVN/Y5INRM.

References

Bestvater, S. E., and B. L. Monroe. 2022. "Sentiment is Not Stance: Target-Aware Opinion Classification for Political Text Analysis." *Political Analysis,* 1–22. https://doi.org/10.1017/pan.2022.10.

Bird, S., E. Klein, and E. Loper. 2009. *Natural Language Processing with Python.* O'Reilly Media Inc. https://www.nltk.org/book/.

DeRobertis, N. 2020. *pysentiment2.* V. 0.1.1. https://github.com/nickderobertis/pysentiment.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* https://doi.org/10.48550/ARXIV.1810.04805.

Goldstein, J. S. 1992. "A Conflict-Cooperation Scale for WEIS Events Data." *The Journal of Conflict Resolution* 36 (2): 369–385. ISSN: 00220027, 15528766.

Häffner, S., M. Hofer, M. Nagl, and J. Walterskirchen. 2023. "Replication Data for: Introducing an Interpretable Deep Learning Approach to Domain-Specific Dictionary Creation: A Use Case for Conflict Prediction." *Harvard Dataverse, V1,* https://doi.org/10.7910/DVN/Y5INRM.

Hooker, S., D. Erhan, P.-J. Kindermans, and B. Kim. 2018. *A Benchmark for Interpretability Methods in Deep Neural Networks.* https://doi.org/10.48550/ARXIV.1806.10758.

Horel, E., V. Mison, T. Xiong, K. Giesecke, and L. Mangu. 2018. *Sensitivity based Neural Networks Explanations.* https://doi.org/10.48550/ARXIV.1812.01029.

Hornik, K., M. Stinchcombe, and H. White. 1989. "Multilayer feedforward networks are universal approximators." *Neural networks* 2 (5): 359–366.

Hu, Y., M. S. Hosseini, E. Skorupa Parolin, J. Osorio, L. Khan, P. Brandt, and V. D'Orazio. 2022. "ConfliBERT: A Pre-trained Language Model for Political Conflict and Violence." In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

Hutto, C., and E. Gilbert. 2014. "Vader: A parsimonious rule-based model for sentiment analysis of social media text." In *Proceedings of the international AAAI conference on web and social media,* 8:216–225. 1.

Khadjeh Nassirtoussi, A., S. Aghabozorgi, T. Ying Wah, and D. C. L. Ngo. 2014. "Text mining for market prediction: A systematic review." *Expert Systems with Applications* 41 (16): 7653–7670. ISSN: 0957-4174. https://doi.org/10.1016/j.eswa.2014.06.009.

Laver, M., K. Benoit, and J. Garry. 2003. "Extracting Policy Positions from Political Texts Using Words as Data." *The American Political Science Review* 97 (2): 311–331. ISSN: 0003-0554.

Lee, J., R. Tang, and J. Lin. 2019. *What Would Elsa Do? Freezing Layers During Transformer Fine-Tuning.* https://doi.org/10.48550/ARXIV.1911.03090.

Lipovetsky, S., and M. Conklin. 2001. "Analysis of regression in game theory approach." *Applied Stochastic Models in Business and Industry* 17 (4): 319–330.

Lo, J., S.-O. Proksch, and J. B. Slapin. 2016. "Ideological Clarity in Multiparty Competition: A New Measure and Test Using Election Manifestos." *British Journal of Political Science* 46 (3): 591–610. https://doi.org/10.1017/S0007123414000192.

Loughran, T., and B. McDonald. 2011. "When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks." *The Journal of finance* 66 (1): 35–65.

Pestov, V. 2013. "Is the k-NN classifier in high dimensions affected by the curse of dimensionality?" Grasping Complexity, *Computers & Mathematics with Applications* 65 (10): 1427–1437. ISSN: 0898-1221. https://doi.org/10.1016/j.camwa.2012.09.011.

Ribeiro, M. T., S. Singh, and C. Guestrin. 2016. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 1135–1144. KDD '16. San Francisco, California, USA: Association for Computing Machinery. ISBN: 9781450342322. https://doi.org/10.1145/2939672.2939778.

Samek, W., A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller. 2016. "Evaluating the visualization of what a deep neural network has learned." *IEEE transactions on neural networks and learning systems* 28 (11): 2660–2673.