Detection of Propaganda Techniques in News Articles

Hemil Desai
hemil10@ucla.edu

Revanth Yamani
revanthky@ucla.edu

Brian Wang
wangbri1@g.ucla.edu

Atharv Sakhala
asakhala@ucla.edu

Abstract

Motivated by the hope of unbiased, unaltered news reporting around the world, we address the question of detecting and classifying propaganda techniques in written text. Here we present our models and results aimed at solving SemEval 2020 Task 11: "Detection of Propaganda Techniques in News Articles". Our approach involves the use of the Hugging Face Transformers Library for each of the two tasks. We tested these models on a provided training corpus of news articles labeled with classifications of spans and techniques, and then evaluated generality and performance on the test set by submitting our prediction models to the competition leaderboard. We compare the performance of multiple transformer models and submitted each to the competition for evaluation. Additionally, we created a web-based application that could host our models and run predictions on input text so users are able to visualize the prediction of propaganda spans and techniques for any text article they provide as well as for included examples. In both tasks, our models outperformed the baseline. Using the Bert model architecture performed the best for Span Identification (0.153 F1), while our results for Technique Classification (0.677 F1) were significantly higher with the highest performing DeBerta-large model.

1 Introduction

Propaganda is a big concern in news reporting around the world. Free and uncensored press is important for disseminating truthful information and preserving the fair exchange of ideas. Examples have been seen throughout history of state governments and organizations attempting to bolster a particular agenda or narrative regarding specific events that may not be entirely accurate. In the past, visual images have been effective to bias consumers and evoke a desired response from a specific audience. Now, as text in the form of news and event reporting is ubiquitous in print and on the Internet, it can be difficult to accurately detect the emotional language and logical fallacies that indicate the presence of propaganda. This is when propaganda is most powerful, and is a relevant issue for the news reaching consumers today. The goal of SemEval 2020 Task 11, titled "Detection of Propaganda Techniques in News Articles", was to find solutions for two separate tasks related to propaganda detection. The first, Span Identification (SI), focused on searching a document for fragments of text containing any type of propaganda. The second, Technique Classification (TC), involves classifying spans of text already identified as propaganda into one of fourteen classes.

We compared the performance of various models found in the Hugging Face Transformers Library to collect results for this task. Here we used models such as Bert, RoBerta, and DeBerta to see if their different implementations would affect overall performance. Our results showed that each model was able to perform the human baseline predictions. The Bert model architecture performed the best for the SI task, and the DeBerta model performed the best for the TC task. The Transformer-based models alone did not show significant results for SI (0.153 F1 for the Bert-base model on the test set). On the other hand, the TC task showed impressive prediction results (0.677 F1 score for the DeBerta-large model) that was among the top five submissions for this challenge on the SemEval task page.

2 Span Identification

The first task in detecting propaganda is span identification. This task involves identifying a span of text in a document that contains a propaganda technique. This is a binary sequence tagging task, where the text is broken into tokens. Each token has a binary classification (whether it is propaganda or not).

In project, we explore several pretrained language models to perform this binary sequence tagging. These models simply use an additional linear layer atop the token embeddings from the language models to perform binary classification of propaganda. Though previously we had utilized the codebase from team **aschern**[1] (another group participating in the span identification task), we decided to implement our own span identification models. The method used in **aschern** combines a RoBERTa model with a Conditional Random Field (CRF) layer. In our case, we experiment with 3 pretrained language models with linear classifier layers. The primary goal of this experiment is to determine the capabilities of these language models alone on span identification, without using special features or classifier layers.

2.1 Processing

For span identification, the dataset consists of news articles with a corresponding list of labels describing the beginning and end indices of propaganda spans within a particular article. Once the article is tokenized, each word id must have a corresponding label (whether it is propaganda or not). Thus, the beginning and end indices must correctly align with the binary labels. A similar challenge happens when converting the predicted binary labels back into beginning and end indices, with the added issue that tokenization often removes certain artifacts from the original text (e.g. text accents, extra spaces), affecting the accuracy of indices. Once this data is preprocessed, we train our pretrained language models.

2.2 Pretrained Language Models

We experiment with 3 different pretrained language models: BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and DeBERTa (He et al., 2020). BERT is a bi-directional transformer model trained over unlabelled data. RoBERTa improves upon the training approach of BERT by using dynamic masking of tokens during training, and removing the next sentence prediction objective. Lastly, DeBERTa improves upon both BERT and RoBERTA by using a disentangled attention mechanism, which represents words as both content embeddings and position embeddings (as opposed to being summed into a single vector in BERT), and alters the masked decoder to incorporate absolute positions of words. We expect that in most cases, a model using DeBERTa should outperform models trained with BERT and RoBERTa - this is because the propaganda spans often occur in reference to a subject (e.g. for propaganda involving name-calling, these cases often occur as the subject of a sentence). More importantly, the relative position of these spans may indicate propaganda. For example, ”Jon Stewart calls the viewers a bunch of ’uninformed sheep’”, shows that there is name calling (”uninformed sheep”), which is in reference to the ’viewers’. In this case, ’Jon Stewart’ is explicitly referring to them via the word ’calls’. We believe a model that explicitly encodes the position of different tokens may be able to better identify spans of propaganda text.

3 Technique Classification

Building upon Span Identification, the second task is to identify specific techniques of propaganda in the spans identified earlier. There are around 18 propaganda techniques [2] identified by the task creators. Some spans have multiple propaganda techniques, but such labels had the same combination of techniques. As a result, the task combines this labels and reduces the label set to 14 techniques or classes. This can be presented as a multi-class sequence classification problem. Next, we will look at the details of transforming and processing the spans for sequence classification.

3.1 Adding Context to Spans

The spans in itself are relatively short compared to the news article. As a result, they fail to capture the context around the propaganda. This context is extremely relevant for accurate technique classification. We represent the span, context and article as follows:

- Article $\mathcal{A}$
- Span $s_i \in \mathcal{S}$. $\mathcal{S}$ is the list of propaganda spans from $\mathcal{A}$.
- Left context c_i^l, Right context c_i^r, Full context $\mathcal{C}_i = \{c_i^l, c_i^r\}$ of the span s_i, $|\mathcal{C}_i|$ the length of the context

[1] https://github.com/aschern/semeval2020$_task11$

[2] https://propaganda.qcri.org/annotations/definitions.html

We represent a sequence for the TC task using two techniques inspired from Aschern (Chernyavskiy et al., 2020) and Applica AI (Jurkiewicz et al., 2020). For our initial approach, we construct the sequence as $[CLS]\ s_i\ [SEP]\ \{c_i^l, s_i, c_i^r\}$. This includes the span twice, and may add noise to the sequence. Also, since the span is included twice, the context length is shortened since we set a limit on the max sequence length for the task. For our preferred approach, we construct the sequence as $[CLS]\ \{c_i^l, [BOP], s_i, [EOP], c_i^r\}$. Here we just combine the left and right context with the span and add special tokens before and after the span (Beginning of Propaganda (BOP) and End of Propaganda (EOP)) to separate it out from the context. Initial runs showed that the second scheme worked much better so we fix on it for our experiments.

For the construction itself, we set a maximum length limit on the entire sequence. This max length is enforced on the number of words. The words are obtained using a basic tokenizer (split by space). The context is constructed dynamically given the sequence length and the span length. First, the remaining length left for context is calculated and equally divided among the left and right contexts. Second, we check if there's any buffer left after adding the left and right contexts. For example, if the span is at the beginning of the article, there's no left context and the buffer will be half of the remaining length. The buffer is then used to add additional context on either side. In our example, we will use the buffer to extend the right context. This is our **dynamic context generation** approach.

We experiment with sequence lengths of 128 and 256. The model's tokenizer increases the sequence length and the final sequence length is constrained in most transformer models, so we set the maximum limit to 256. Going shorter than 128 didn't make sense cause that would result in minimal context.

3.2 Model Architecture

Once we have the sequence from the section above, we pass it through a transformer backbone. We experiment with 4 different transformer backbones - BERT (Devlin et al., 2018), DistilBERT (Sanh et al., 2019), RoBERTa (Liu et al., 2019), and DeBERTa (He et al., 2020). The details of three of them are described in Section 2.2. DistilBERT is a compressed version of BERT obtained from distillation. It is smaller in size allowing for faster infernce but at the cost of some performance drop.

A pass through the transformer backbone gives the output representations for each token in the sequence. We take the output representations from the [CLS] token and pass it through a Dropout layer (Srivastava et al., 2014) followed by a Linear layer. The architecture is summarized in Fig. 1.

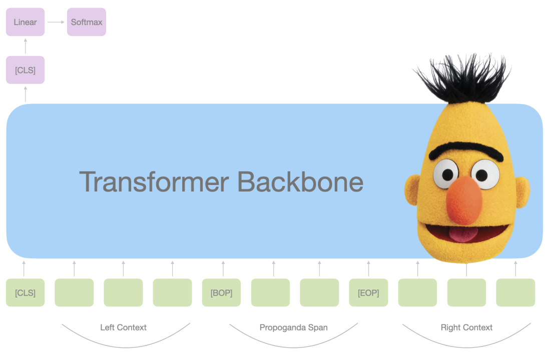


Figure 1: Technique Classification Model Architecture

4 Results

For all our experiments, we use Huggingface Transformers (Wolf et al., 2020) for implementation. The code is available in our github repo [3].

4.1 Span Identification

In all experiments, we trained each model using the AdamW optimizer with a learning rate of 5e-5. They are trained with batch sizes of 16 (with the exception of DeBERTa, which used a batch size of 4 due to memory limits) , and for 25 epochs.

We first evaluate the performance of our models on the unseen dev set. We also test **aschern's** models, which utilize a CRF instead of a linear classifier. We evaluate performance by the F1, precision and recall values for the predicted binary labels and the ground truth.

The results are shown in Table 1. One common factor between all of our models is that they exhibit high precision but low recall. It's possible that our models are overfitting to a set of propaganda examples, and only classify this known set of examples as propaganda. This problem is prevalent in all models submitted to this competition on the test set, though to a lesser degree. The table also shows that **Aschern** significantly outperforms our models,

[3]`https://github.com/asakhala921/Propoganda-Detection-in-News-Articles`

Model	**F1**	Precision	Recall
bert-base (ours)	0.394	0.804	0.145
roberta-base (ours)	0.228	0.555	0.270
deberta-base (ours)	0.232	0.598	0.300
bert-CRF	0.45	0.43	0.48
distilbert-CRF	0.43	0.42	0.45
roberta-CRF	0.46	0.42	0.51

Table 1: Span Identification performance on dev set

Model	**F1**	Precision	Recall
bert-base-uncased	**0.153**	**0.369**	**0.096**
roberta-base	0.111	0.298	0.068
baseline	0.0032	0.13	0.00162

Table 2: Span Identification performance on test set

signifying the importance of having a CRF layer at the end of our models. This is likely due to the improved ability of CRF's to model this structured prediction problem, rather than viewing each token classification as independent.

To measure our performance on the test set, we uploaded our predictions to the competition website[4]. The results are shown in Table 2. We omitted the results for DeBERTa, as it performed much worse in all cases. The baseline for the competition is a random guesser for determining propaganda spans. Similar to the dev set results, precision is often much higher than recall - more than likely, our models have poor generalizability to new examples. In particular, we notice that the models generate fewer positive predictions for propaganda in the cases of RoBERTa and DeBERTa.

Another interesting note is that DeBERTa seems to perform worse than the other models. In addition to improved training approaches and model architectures, RoBERTa and DeBERTa should have an advantage due to being trained on news datasets. Though BERT enjoys none of these advantages, it still outperforms the others on both the dev and test dataset. Why this is the case, we do not know the exact answer for this reason and more testing would allow us to explore this issue further.

[4]https://propaganda.qcri.org/

Transformer model	**Sequence length**	**Micro f-1 score**
bert-base-uncased	256	0.6124
distillbert-base-uncased	256	0.5970
roberta-base	256	0.6359
deberta-base	256	0.6557
deberta-large	256	0.6773
bert-base-uncased	128	0.6209
distillbert-base-uncased	128	0.5936
roberta-base	128	0.6256
deberta-base	128	0.6341

Table 3: Technique Classification performance on dev set

4.2 Technique Classification

We trained our models with a learning rate of 5e-5 and with a weight decay of 1e-4. We used a total batch size of 8 and trained till 3 to 5 epochs (depending on the size of the model). We limit our batch sizes per GPU to 2 and use 4 gradient accumulation steps to fit within the constraints of the GPU. For the learning rate, we limit warm-up steps to a 100.

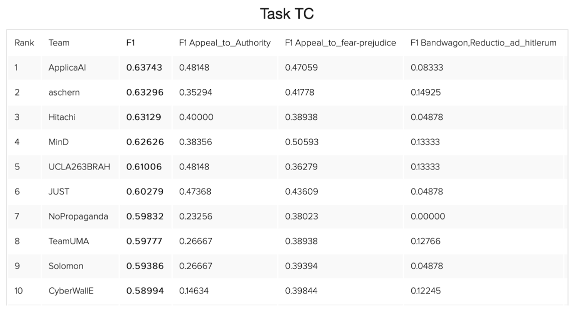
We run our experiments for total sequence lengths (as described in Section 3.1) of 128 and 256 and train various models. Our results are shown in Table 3.

We can see that the best performing model is Deberta, which makes sense given its novel update to the attention architecture. We also see that increasing the context length (via increasing the max sequence length) usually improves performance.

Since Deberta performs the best on our base experiments, we train deberta-large for our final model.It gives us a micro F1 score of 0.6773 on the development set. We train it with both train and development sets combined, and submit this to the task leaderboard and use it as our final model for inference in the web application too. The model secures us position 5 on the SemEval leaderboard for the TC task. (See Figure 2). It is available for download at `https://huggingface.co/hd10/semeval2020_task11_tc`.

Our best accuracy is on the technique class of ”Loaded Language”, while the worst performing class is ”Black-and-White Fallacy”.

We had to truncate many of our experiments due to memory and time constraints. We used FP16 training whenever we ran into GPU memory issues. We expect to see higher gains and more experiments with access to a higher compute budget.



Task TC

Rank	Team	F1	F1 Appeal_to_Authority	F1 Appeal_to_fear-prejudice	F1 Bandwagon,Reductio_ad_hitlerum
1	ApplicaAI	**0.63743**	0.48148	0.47059	0.08333
2	aschern	**0.63296**	0.35294	0.41778	0.14925
3	Hitachi	**0.63129**	0.40000	0.38938	0.04878
4	MinD	**0.62626**	0.38356	0.50593	0.13333
5	UCLA263BRAH	**0.61006**	0.48148	0.36279	0.13333
6	JUST	**0.60279**	0.47368	0.43609	0.04878
7	NoPropaganda	**0.59832**	0.23256	0.38023	0.00000
8	TeamUMA	**0.59777**	0.26667	0.38938	0.12766
9	Solomon	**0.59386**	0.26667	0.39394	0.04878
10	CyberWallE	**0.58994**	0.14634	0.39844	0.12245

Figure 2: Technique Classification Task Leaderboard

5 Web Application

We also built a web application to visualize the end to end inference pipeline starting from a news article all the way to technique classification. The application takes an article or sentence as input, and then runs the following pipeline

- Inference using the Span Identification model to generate spans containing propoganda
- Data pre-processing on the generated spans from the above task
- Inference using the Technique Classification model on the processed spans

The articles can currently be added as input to the application using a text box, and the pipeline can be initiated from scratch for each new article. The application currently supports caching of old articles so as to provide instant results if a user wants to revisit the old article. The application also has a sidebar with feedback sliders for both SI and TC tasks. These sliders are a way to collect quick and general feedback on how the model performed on the article. This feedback from the user can be used as weak supervision to enhance the dataset and support active learning pipelines. For instance, this application can be used as a labeling function in Snorkel (Ratner et al., 2017). We use Streamlit (str) and Spacy (Honnibal et al., 2020) for developing the application. We currently serve it using Cloudflared tunnels [5] but the application can be hosted using other avenues. The application overview is represented in Figure. 3.

[5]https://github.com/cloudflare/cloudflared

6 Future Work

In the future, we want to explore why BERT seems to outperform the more recent pretrained language models on span identification. In addition, we can experiment with different classifier layers, such as Conditional Random Fields, as the span identification task involves some structured prediction. Hyperparameter tuning such as epoch and batch sizes could also be an area of additional work that we identified during our experiments. We would also like to explore ensemble methods of prediction that combine distinct architectures or models into a single classifier that we believe could provide new insights and very strong results for this challenge, in both span identification and technique classification.

References

Streamlit - the fastest way to build and share data apps.

Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. 2020. aschern at semeval-2020 task 11: It takes three to tango: Roberta, crf, and transfer learning. *arXiv preprint arXiv:2008.02837*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. Applicaai at semeval-2020 task 11: On roberta-crf, span cls and whether self-training helps them. *arXiv preprint arXiv:2005.07934*.

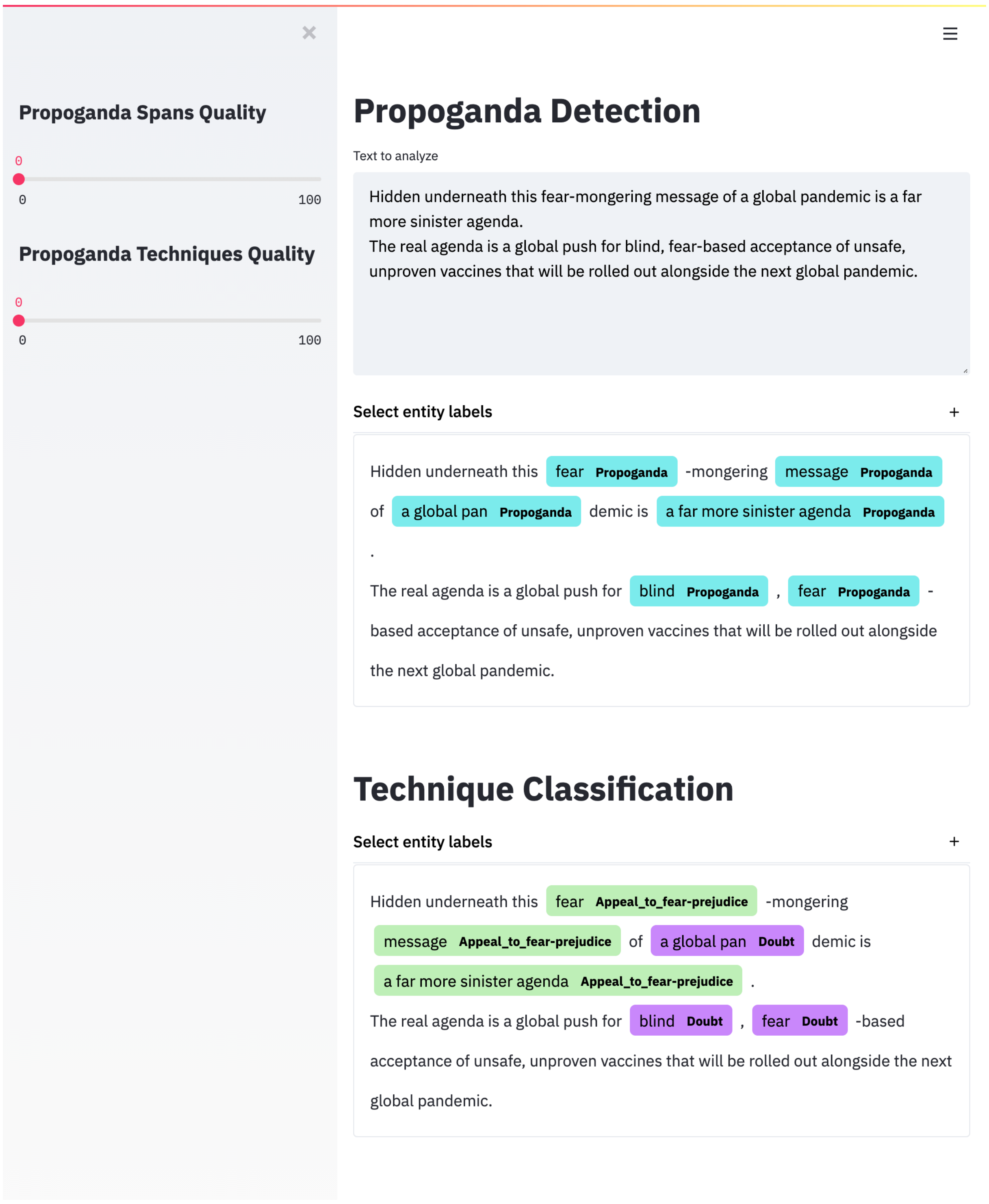
Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.


Propoganda Spans Quality

0

0 100

Propoganda Techniques Quality

0

0 100

Propoganda Detection

Text to analyze

Hidden underneath this fear-mongering message of a global pandemic is a far more sinister agenda.
The real agenda is a global push for blind, fear-based acceptance of unsafe, unproven vaccines that will be rolled out alongside the next global pandemic.

Select entity labels

Hidden underneath this fear **Propoganda** -mongering message **Propoganda** of a global pan **Propoganda** demic is a far more sinister agenda **Propoganda** .
The real agenda is a global push for blind **Propoganda** , fear **Propoganda** -based acceptance of unsafe, unproven vaccines that will be rolled out alongside the next global pandemic.

Technique Classification

Select entity labels

Hidden underneath this fear **Appeal_to_fear-prejudice** -mongering message **Appeal_to_fear-prejudice** of a global pan **Doubt** demic is a far more sinister agenda **Appeal_to_fear-prejudice** .
The real agenda is a global push for blind **Doubt** , fear **Doubt** -based acceptance of unsafe, unproven vaccines that will be rolled out alongside the next global pandemic.

Figure 3: Web Application Overview