ELECTION CAMPAIGN DYNAMICS

A PREPRINT

Emre CANBAZER * †
emre.canbazer5
@etu.univ-lorraine.fr

Melpomeni CHIOUTAKOU * †
melpomeni.chioutakou-chioutakakou6
@etu.univ-lorraine.fr

Amir Hossein JAMALI * †
amir-hossein.jamali7
@etu.univ-lorraine.fr

Anahita POULAD * †
anahita.poulad
@etu.univ-lorraine.fr

February 7, 2022

ABSTRACT

This report introduces the website developed to provide real-time election campaign corpora for the upcoming 2022 presidential elections in France, extracted from social media of the declared candidates. Designed for the use of researchers from non-technical domains, the website is envisaged to also provide insights of the retrieved corpora in terms of topic relations and similarity. The prototype of the website[3] is online and can be consulted. We present the process and challenges of creating this tool. The code of this work can be accesssed on our GitHub repository[4].

1 Introduction

The pre-election period is an intense period where political campaigns take the reins of everyday life. During this period, arguments, ideas and topics flow between the candidates and one can wonder how the discourse of a campaign is constructed and want to understand the main campaign focus points, its key topics, their life-cycle and how they evolve during campaigns. Political speeches, during campaign season are known for being formulaic and repetitive. What could we learn by looking for patterns in the way the speeches change over the course of the campaign? This report

*Equal contribution, alphabetically ordered

†Msc Natural Language Processing, IDMC - University of Lorraine, Nancy, France

[3] https://france2022.azurewebsites.net/

[4] https://github.com/amirpaia/election-campaign-dynamics

focuses on this exact question and tries to provide a tool for the extraction of political corpora and their analysis. It hopes to serve as a tool for discovering the evolution of the campaigns of the presidential elections in France, in 2022.

In the following sections, we present our methodology in detail.

2 Data gathering

Since the aim of our work is to provide corpora for researchers from non-technical domains, the data that we create should not be overly complex to use and hard to read. Thus, we decided to provide tabular data in .CSV format since it is easy to inspect and manipulate on Python's pandas as well as on Microsoft Excel. We have scripted two programs to create two different corpora; one for Twitter data and one for YouTube data. These scripts are used within the back-end development of the website. However, they can also be used on their own and they can be found in Jupyter Notebook format within our GitHub repository.

2.1 Twitter Data

The Twitter corpus is created using the Twitter API service provided by Twitter Developers. Using the credential keys assigned to this project by Twitter Developers, we could use the tweepy package to import the data of a given user in JSON format. We decided what information to include in the dataframe other than the preliminary information such as the name, profile bio and the tweet content based on our experiments with topic modeling and the correspondences that we made with the testers of the corpora. The features present in the dataframe that the program returns are as follows:

- **name**: (str) The screen name of the user. It is important to note that this may not be the exact full name of the politician. We assume that the extra information on the name is not going to be used for the tasks since we have only encountered a few examples of a french flag and the hashtag *#presidentielle2022* which are not distinctive.
- **bio**: (str) The Twitter bio of the user. This information can be used alongside the tweet texts for exploratory tasks, however, in case of politicians, they tend to be rather short.
- **follower_count**: (int) The number of followers of the account. This is a crucial data since it shows how much influence the politician has.
- **tweet_text**: (str) The whole tweet of the user.
- **tweet_id**: (int) The id of the tweet on Twitter, one can use this id to access the tweet on twitter.com.

- **fav_count**: (int) The number of likes (formerly 'favorites') that the tweet got so far. This and the next feature is important for the analysis of the influence of the tweet.
- **rt_count**: (int) The number of retweets the tweet has received so far, ie. the number of users who shared the tweet on their own account.
- **in_reply_to**: (str, None) If the tweet is a reply to a user, it shows what user it is a reply to, if not, the value wilşl be None. Also, if the value is the same as the author of the tweet, it means that it is a 'thread' ie. the author is adding to their own tweet.
- **mentions**: (list, None) The list of user ID's that are mentioned (ie. 'tagged') within the tweet. The value is None if no user is mentioned. Moreover, if the tweet is a retweet, the first user in the list is the user that the tweet is retweeted from.
- **is_retweet**: (boolean) This feature demonstrates whether the tweet is a retweet (ie. shared from anoter user). Thus, if True, the author is not the same person as the queried politician.
- **is_quote**: (boolean) This shows whether the tweet is a 'quote' or not. If True, the tweet is referencing another tweet.
- **hashtags**: (list, None) The list of #hashtags within the tweet. It is important to preserve them since they get removed by most of the preprocessing functions and they provide crucial since they function as keywords that are deliberately given by the author. The value will be None if there is no hashtags in the tweet.
- **emojicons**: (list, None) The list of 'wordified' versions of the emojis present in the tweet. For instance, for '🇫🇷', the value will be *flag_france*. Just like the previous feature, this one also gets lost during the preprocessing although it contains important information especially for sentiment analysis tasks. This process is done using the emoji [5] package for Python. The value will be None if no emoji is used in the tweet.
- **time**: (pandas timestamp) The exact time that the tweet was published. We use this information for performing topic modeling over time that we will explain in section 3.

2.2 YouTube Data

The YouTube corpus that we provide is more complex and less practical to use than the Twitter corpus. Nevertheless, we still think that it is necessary to have a tool that creates a transcribed speech corpus to analyse political campaign speech. We currently provide a YouTube corpora generation program that outputs .CSV dataframe with automatically generated captions provided by

[5] `https://pypi.org/project/emoji/`

YouTube, split into one-minute time intervals. We chose the automatically-generated captions due to the sparsity of human-transcribed captions added to the videos. We are aware that the output is not necessarily a perfectly well-formed text every time, but according to our experience, they are good enough to be used for bag-of-words tasks. We will be explaining some of our experimentation on this dataset in section 3. The features that the generated dataframe are as follows:

- **channel_title**: (str) The title of the YouTube channel that publishes the video, for reference.
- **channel_id**: (str) The ID of the YouTube channel that publish the video, one can have access to the YouTube account with this ID.
- **video_title**: (str) The title of the video that is been queried.
- **video_description**: (str) The description that is written by the publisher of the video. For the videos of campaign speeches, this is important because it usually contains crucial information such as the orator, the time and place of the speech being given.
- **upload_date**: (datetime object) The date and time of the publishing of the video. It is important to note that this date is not necessarily the date of the speech.
- **duration**: (str) The total duration of the video in *hour-minute-second*.
- **time_caption**: (datetime object) The time of the generated captions, split into one-minute intervals. Thus, one row represents one minute of speech.
- **transcript**: (str) The transcript of the speech taken form automatically generated captions of the video.

The first six features of the dataframe, which contains the metadata of the speech, is extracted using the YouTube API (v3) service provided by Google Developers whereas the last two features are extracted using the youtube-transcript-api [6] package for Python.

3 Dynamic Topic modeling of the Campaigns

In order to capture the main ideas of the campaigns by analyzing the direct text produced by the candidates, we have relied on the candidates' tweet corpora, since Twitter constitutes nowadays the most direct informative social network. Our work was focused on the recurring topics that appear in the text generated by the candidates, since highlighting these topics could serve as an effective feature in analyzing the main ideas discussed in an election campaign.

[6] `https://pypi.org/project/youtube-transcript-api/`

Topic modeling is an unsupervised machine learning technique used to retrieve the different topics that are present in a document. In our case, we leveraged this technique in favor of our hope to provide relevant information to a non-technical audience about the presidential campaign and the different topics invoked by the candidates.

Dynamic topic modeling with BERTopic (Grootendorst 2020a), a robust approach to understanding how topics evolve over time, was used to identify the main topics of each candidate's campaign and their development over the period of these campaigns, while keyword extraction with KeyBERT(Grootendorst 2020b) was added as an additional step to reduce the time it takes for the BERTopic model to be built and to retrieve more meaningful topics.

3.1 Dynamic topic modeling with BERTopic

BERTopic is a topic modeling technique that uses BERT embeddings and c-TF-IDF to create dense, informative clusters which can provide easily interpretable topics and keep important words in the topic descriptions. A few other great advantages it possesses are its interactive visualization ability and its accessibility.

The BERTopic algorithm contains three stages. First, it extracts document embeddings with Sentence Transformers. By default, it uses "paraphrase-MiniLM-L6-v2" - an English BERT-based model trained specifically for semantic similarity tasks- and "paraphrase-multilingual-MiniLM-L12-v2" (Reimers and Gurevych 2019) - similar to the first, but it works for 50+ languages, allowing us to perform BERTopic on French data. At the second step, it clusters documents with the use of UMAP to reduce the dimensionality of embeddings and the HDBSCAN technique to create clusters of semantically similar documents. The last step consists in creating Topic Representation. It extracts and reduces topics with class-based TF-IDF (c-TF-IDF) and then improves the coherence of the words with Maximal Marginal Relevance (MMR).

3.2 Keyword extraction with KeyBERT

KeyBERT is a simple keyword extraction algorithm that uses BERT embeddings to generate keywords and key phrases from a document based on their similarity to the document. First, document embeddings are extracted with BERT to get a document-level representation. Next, the embeddings of words are extracted for N-gram phrases. The similarity of each keyphrase to the document is then measured using cosine similarity. The most similar words can then be identified as the words that best describe the entire document and are considered as keywords.

The implementation of KeyBERT before BERTopic modeling, rendered our topic modeling less time-consuming and helped in obtaining more meaningful results.

3.3 Preprocessing

In order to create a BERTopic object in Python and proceed to dynamic topic modeling, a preprocessed list of documents is needed as well as the time and date. The preprocessing of the tweet data was performed before applying Keyword extraction with KeyBERT. The BERTopic models of the candidates were then built based on these preprocessed lists of keywords.

The dataset was preprocessed as follows. Re-tweets, http/bitly links, hashtags, punctuation, double spacing, numbers, photos, emoticons, stopwords and verbs were removed and tokens were lemmatized. We proceeded in the removal of verbs since they do not help in the discovery of more interpretable topics. The lemmatization and removal of verbs was achieved with "SpaCy" (Honnibal and Montani 2017), whose lemmatization of French corpora performs better than other libraries used.

3.4 Topics over time

In the figures below, BERTopic's "Topics over Time" visualization is presented. This method is interactive and it is possible to retrieve the most relevant keywords of a topic for a particular date over the given period.

If the Topic representation does not yield good results for some topics, it is always possible to extend the stopword list with the words we would like to remove and build the model again using the new preprocessed dataset. For example, in Figure 2 below, we could remove the words "ben" and "bravo", seen in topic 5 and do the procedure again, but we preferred to showcase the original results.

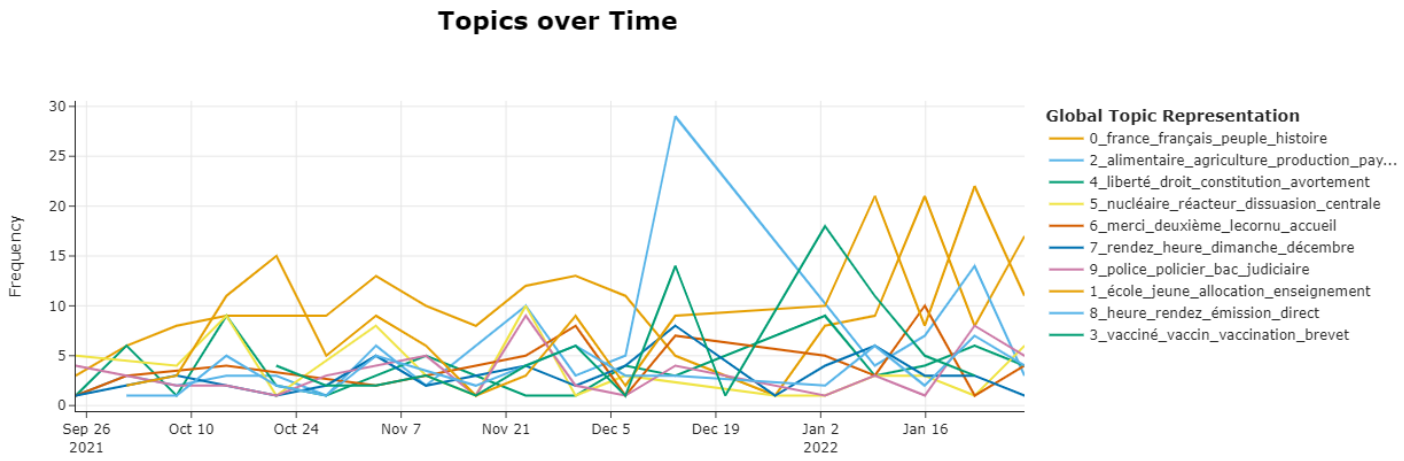

Figure 1: Jean-Luc Mélenchon's 10 most frequent topics on Twitter over the period of September 1st 2021 to February 5th 2022.

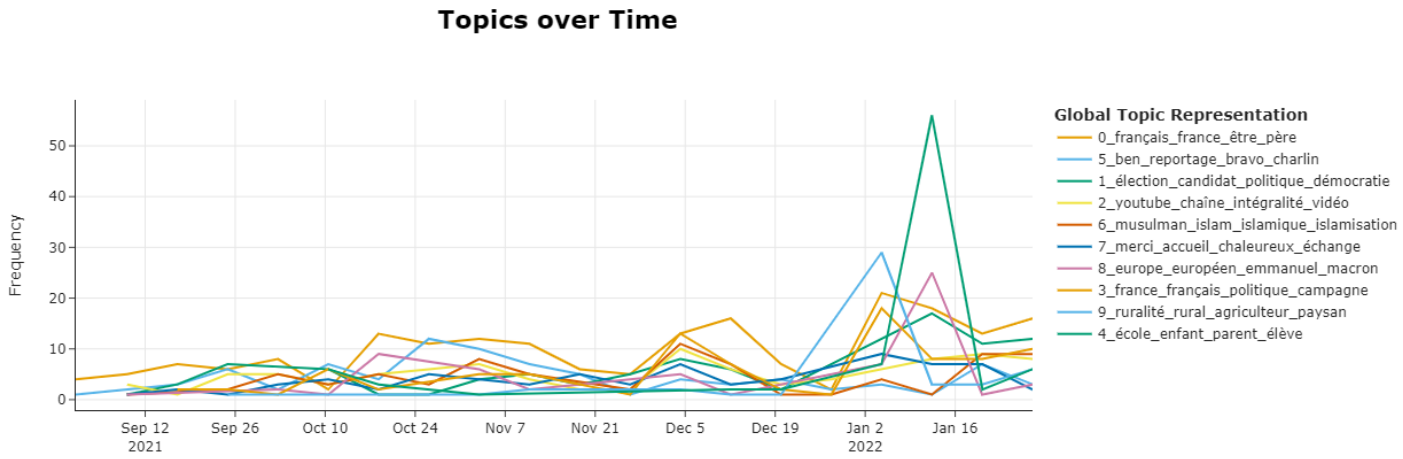

Figure 2: Éric Zemmour's 10 most frequent topics on Twitter over the period of September 1st 2021 to February 5th 2022.

4 Topic Relation

In addition to obtaining the topics of each individual candidate, we wanted to see how these topics interact, i.e. how pairs of candidates interact based on their most frequent topics retrieved.

For that purpose, we explored the semantic similarity between the names of the topics, in this case the four words that denote each topic.

Similarity is determined by comparing word vectors or "word embeddings", multi-dimensional meaning representations of a word. SpaCy's pipeline provides a *similarity* method which makes this task rather easy.

In the table below we present the similarity between the topics seen in the previous section (Figures 1 and 2).

Overall, based on the most frequent topics and their semantic similarity, it seems that Jean-Luc Mélenchon and Éric Zemmour, two candidates from opposing political camps evoke some of the same topics.

However, there are downsides to these methods. Computing similarity scores can be helpful but as thoroughly explained in SpaCy's documentation, a single "similarity" score will always be a mix of different signals, and vectors trained on different data can produce very different results that may not be useful. In Table 1, a high similarity is correctly assigned between topics "0_france_français_peuple_histoire" and "3_france_français_politique_campagne", as well as "1_école_jeune_allocation_enseignement" and "4_école_enfant_parent_élève" but a

	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
Topic 0	0.688	0.636	0.277	0.84	0.394	0.0456	0.728	0.289	0.563	0.566
Topic 1	0.597	0.564	0.253	0.595	0.783	-0.058	0.457	0.373	0.307	0.509
Topic 2	0.269	0.252	0.404	0.404	0.148	0.181	0.224	0.323	0.198	0.37
Topic 3	0.355	0.383	0.0838	0.3	0.293	-0.0678	0.272	0.15	0.224	0.236
Topic 4	0.5	0.714	0.193	0.632	0.385	-0.109	0.656	0.289	0.413	0.452
Topic 5	0.445	0.54	0.262	0.55	0.27	-0.108	0.502	0.199	0.414	0.448
Topic 6	0.187	0.179	0.178	0.21	0.28	0.355	0.0906	0.848	0.149	0.178
Topic 7	0.181	0.217	0.149	0.204	0.211	0.0475	0.0684	0.324	0.206	0.0876
Topic 8	0.296	0.329	0.495	0.367	0.269	0.111	0.202	0.444	0.257	0.175
Topic 9	0.377	0.424	0.113	0.354	0.345	0.0359	0.279	0.13	0.178	0.268

Table 1: Topic relation over the 10 most frequent topics that emerged from the campaigns of two candidates, Jean-Luc Mélenchon (on the left) and Éric Zemmour (on top) over the period of September 1st 2021 to February 5th 2022. Topic 0 represents the most frequent topic.

low similarity score is observed between topics "2_alimentaire_agriculture_production_pay" and "9_ruralité_rural_agriculteur_paysan" when it should be a lot higher.

What is also interesting to mention about this table, is the high correlation between topic "4_liberté_droit_constitution_avortement" and topic "1_élection_candidat_politique_démocratie" (0.714). It is understandable why the similarity method gives a high score between these two but the keywords do not necessarily reflect a similar stance. Since BERTopic allows us to retrieve the words most present in each topic, we can explore these two topics a little more. The words that comprise Topic 4 are: ["liberté", "droit", "constitution", "avortement", "genre", "ivg", "fondamental", "république", "autonomie", "dignité"], while for Topic 2: ["élection", "candidat", "politique", "démocratie", "politicien", "présidentiel", "droite", "électeur", "idée", "autre"]. The topics become immediately clearer.

5 Evaluation

Evaluating a topic model is always a challenging task, since topic modeling is an unsupervised task in nature and we do not have access to the ground truth while building the model. Actually even the existence of such a ground truth in case of topic modeling could be doubted. Human evaluators could find different topics as the main and most important topic mentioned in any document, considering the fact that finding the main and most important topics is a subjective task to some extent. To the best of our knowledge every standard measure or evaluation tool that has been offered thus far for assessment of the topic modeling models, has serious limitations and drawbacks.

We decided to evaluate our model according to the topic coherence aspect and in doing so, we used the Normalized Pointwise Mutual Information (NPMI). This metric reflects the topic coherence between most relevant words in each topic. NPMI ranges from [-1,1] and it represents how much the top words of a topic are related to each other, where higher positive NMPI denotes a better coherence in the topic found by the model. In our work the NPMI was implemented using the gensim library. In our implementation the Countvectorizer in Bertopic is the same as what we used to create the dictionary, corpus and tokens.

We tried accessing the Countvectorizer directly in Bertopic. That way, we did not have to create different instances that might not match exactly.

It is important to stress that metrics such as cv and NPMI are merely proxies for a topic model's performance. They are by no means a ground truth and can have significant issues (e.g., sensitive to the number of words in a topic).

These are the results that we obtained:

Anne Hidalgo : 0.676
Yannick Jadot : 0.698
Marine Le Pen : 0.668
Eric Zemmour : 0.693
JLMelenchon : 0.723
EmmanuelMacron : 0.746

We decided not to use this coherence results to fine tune our models We are very hesitant when it comes to recommending a coherence metric to use as a tuning guide. It is possible to quickly overfit on such a metric when tuning the parameters of BERTopic (or any other topic modeling technique) which in practice might result in poor performance. In other words, we would like to prevent users from solely focusing on grid-searching parameters and motivate them to look at the results.

Having said that, that does not mean that these metrics cannot be used. They are extremely useful under the right circumstances. When we need to compare topic models, these kinds of metrics (e.g., npmi) can definitely be used but we need to take certain things into account. For example, both models need to have the same number of topics and the same number of words need to be in those topics.

6 Website

6.1 Development

Our goal as said is to provide information for researchers and citizens in a simple, usable way. To this end, a website has been developed. To develop the server side, we used Flask as a framework that uses Python as its language. In the client side, Bootstrap and jQuery, as standard frameworks and libraries, are being used. It allowed us to implement a simple, clean, and responsive website in a short period of time. As previously mentioned in section 2, some python libraries such as *tweepy*, and *youtube-transcript-api* are being used to download tweets information from Twitter and video captions from Youtube. Moreover, *nltk* and *spaCy* are being used for preprocessing. In addition, we were supposed to take advantage of *Bertopic* as the main tool to performed topic modeling.

6.2 Deployment and challenges

There are multiple methods and a set of platforms that can be used in order to deploy a website. Considering multiple factors such as security, flexibility, supporting different frameworks and programming languages, and a good management portal, we concluded to Azure.

To be able to deploy a Flask application on Azure, a set of configurations is needed on the application itself, and on the Azure portal as well. One of the most important configuration on the application is to create a proper docker file. The specifications in the docker file begins with the type of OS, the specific version of Python, and a command to copy a number of important files. Then it continues with a command to install required Python packages on the container, and it finishes with setting the startup function of the Flask application.

The other necessary configurations should be done on the Azure portal. First, a *Resource Group* as a logical container is going to be made, then a *Web App* should be created and assigned to that *Resource Group*. By having this container ready, there is the possibility to push a docker image on the server. However, Azure provides a private registry service, namely *Azure Container Registry*, that builds, stores, and manages container images and its artifacts. We created an *ACR* to be able to push different docker images into Azure with different tags. This enables us to switch between different versions of the application if it is needed.

Although everything could be handled during the deployment process, it was not a task as simple and straightforward as it may seem. There was an inevitable package conflict due to the many Python packages that have been used in this application to develop topic modeling and its visualization. A number of conflicts emerged when installing a different version of packages and replacing them by another package that provides the same abilities. There was a conflict with BERTopic and more specifically its Visualization over time method, the core of our system, that

prevents us from deploying the final version of the website on the server. While it works on the local machine, it faces problems while running on the server. We will try to fix this bug and provide a complete tool in the near future.

7 Discussion & Further Improvements

In this section, we share our thoughts on how our methodology could potentially be improved, based on sifferent ideas and encountered problems.

The corpora that our website provides are open to improvement. Firstly, the time of creating the Twitter corpora could be optimized; since tweepy discontinued the method to extract only the tweets that are in a certain time period, our program has to fetch all the tweets that the queried user has posted so far, and then filters them according to the time period that is selected by the user. This results in a certain time loss, especially if the selected time period is rather short. We will be checking for any updates from the side of Twitter API to solve this. Another possible improvement on the corpora is the transcript of speech within the YouTube corpora. As we mentioned in section 2, we rely on YouTube's speech-to-text technology at this point of the project. We acknowledge that the question of the quality of the generated text is an open research question. However, as we assume that the the text is not perfect, we are curious to improve the quality of the generated text by passing the text through a pipeline that checks the spelling, coherence and cohesion.

Concerning the analysis of the campaigns, a more impactful way to analyze the topics would be through graphical representations that present the importance and composition of topics over time. Those are being finalized and can be provided for further analysis.

A plausible next step for the evaluation could be to use a great package for evaluating our topic model, namely OCTIS. It has many evaluation measures implemented aside from the standard coherence metrics, such as topic diversity, similarity, and classification metrics. We would choose a subset of evaluation metrics there that best suits our use case.

8 Acknowledgements

We would like to thank Maxime Amblard, for their initial idea and guidance throughout the project; Fabienne Greffet for trying our website and sharing their feedback with us;and Miguel Couceiro and Esteban Marquer for their suggestions and involvement.

References

Grootendorst, M. (2020a). "BERTopic: Leveraging BERT and c-TF-IDF to create easily interpretable topics." Version v0.9.4. In: DOI: 10.5281/zenodo.4381785. URL: https://doi.org/10.5281/zenodo.4381785.

Grootendorst, M. (2020b). *KeyBERT: Minimal keyword extraction with BERT.* Version v0.3.0. DOI: 10.5281/zenodo.4461265. URL: https://doi.org/10.5281/zenodo.4461265.

Honnibal, M. and I. Montani (2017). "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing". To appear.

Reimers, N. and I. Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. URL: http://arxiv.org/abs/1908.10084.