

Efficient Use of Large Language Models for Analysis of Text Corpora

David Adamczyk¹^a and Jan Hůla^{1,2}^b

¹*Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, Ostrava, 701 03, Czech Republic*

²*Czech Technical University in Prague, Prague, Czechia, Czech Republic*

Keywords: Large Language Models, Embeddings, Word Representations, Sentence Classification, Efficient Pipeline.

Abstract: In this paper, we propose an efficient approach for tracking a given phenomenon in a corpus using natural language processing (NLP) methods. The topic of tracking phenomena in a corpus is important, especially in the fields of sociology, psychology, and economics, which study human behavior in society. Unlike existing approaches that rely on universal large language models (LLMs), which are computationally expensive, we focus on using computationally less expensive methods. These methods allow for high data processing speed while maintaining high accuracy. Our approach is inspired by the cascade approach to optimization, where we first roughly filter out unwanted information and then gradually use more accurate models, which are computationally more expensive. In this way, we are able to process large amounts of data with high accuracy using different models, while also reducing the overall cost of computations. To demonstrate the proposed method, we chose a task that consists of finding the frequency of occurrence of a certain phenomenon in a large text corpus, which is divided into individual months of the year. In practice, this means that we can, for example, use Internet discussions to find out how much people are discussing a particular topic. The entire solution is presented as a pipeline, which consists of individual phases that successively process text data using methods selected to minimize the overall cost of processing all data.

1 INTRODUCTION


In this work, we present an efficient text mining pipeline for large text corpora. Text mining is a process of extracting knowledge from unstructured text data. It can be used to identify patterns and trends in large text corpora, extract keywords and phrases, classify text into different categories, and summarize the content of text documents. It has a wide range of applications, including market research, customer sentiment analysis, and fraud detection.


With the growing popularity of Large Language Models (LLMs), which excel in natural language processing tasks, there is an increasing demand for these applications because new opportunities become available. However, instead of relying solely on LLMs, our aim is to design efficient pipelines that have much lower hardware requirements and create models specialized for a given task. Our work focuses on processing large text corpora of a given domain and we are interested in tracking various phenomena in these corpora. Concretely, we are interested in tracking a

given phenomenon in time.

Such problems may arise in various fields such as sociology, psychology, or economics, where researchers may be interested in analyzing the beliefs or opinions of a population through an analysis of online discussions. For example, one can study how opinions about a given topic change in connection with a given event. It is clear that if an event is very important, people will talk about it online and express their opinions. By analyzing online discussions, we can track the opinions of people and get a better understanding of how they perceive the event.

The essence of our approach is to gradually apply several classification methods in order to effectively balance computational cost and the speed of computation. The task we chose to demonstrate our proposed approach is to calculate the frequencies of a monitored event in individual months. In simple terms, we want to count how often people discuss topics such as political elections or weather in given months. For this purpose, we will use a data set containing comments from social networks, where individual comments are divided by month from 2019 to 2022. A naive approach is to perform inference for each comment using a large language model. Such an approach

^a <https://orcid.org/0000-0002-7794-104X>

^b <https://orcid.org/0000-0001-7639-864X>

is both very expensive and time-consuming. We want to use a combination of techniques, where some excel in high speed and low computational cost but have lower accuracy, while other techniques excel in high accuracy but are computationally very demanding. A cascade of these techniques forms a pipeline that first very cheaply and quickly filters out irrelevant comments and then gradually processes a smaller amount using more accurate techniques so that the entire process is accurate and at the same time fast with the lowest possible computational cost.

2 RELATED WORK

Modern quantitative text analysis in sociology is largely based on research in mass communication that focused on the press and political propaganda in the 1930s and 1940s (Krippendorff, 2018).

In recent years, the development of natural language processing and machine learning has offered new opportunities for using computers for text analysis in the social sciences. Many of these methods are described in (Macanovic, 2022), which divides the main approaches into several categories.

The identification of phenomena in large language corpora (Spörlein and Schlueter, 2021) used a dictionary-based method to analyze text data from YouTube comments on migration-related videos to study the development of ethnic slurs against multiple minority groups during the large influx of refugees to Germany in 2015. Topic modeling is increasingly being used in the humanities, literary studies, history, and also in political science (Törnberg and Törnberg, 2016). Many of these approaches use LDA or Bayesian approaches to infer latent topics in news articles, scientific journals, or blog posts.

There are analyses that focus directly on the use of large language models in the context of their use for the analysis of large amounts of data. Individual analyses can also be thematically focused according to the analyzed domain. The topic of global warming and the use of ChatGPT is addressed by (Biswas, 2023a). Another interesting domain could be the use in healthcare (Biswas, 2023b), (Patel and Lam, 2023). These analyses focus specifically on the use of ChatGPT.

Some users consider language models to be disruptive technologies, and their use is evident not only in academia and education. The purpose of the study (Törnberg, 2023) is to analyze data from the LinkedIn network in the form of user comments about the ChatGPT tool. This study shows that plagiarism, references, citations, and reviews of the liter-

ature are among the topics that raise the greatest concerns among these users.

Another analysis of ChatGPT users from a group of academics is presented in (Bukar et al., 2023). This analysis shows a division of the academic community into a group of enthusiasts and a group that has significant reservations about the use of LLMs. Among other things, it can be inferred from the comments that a large part of the users are very impressed with ChatGPT's performance and its potential to help with activities related to research, data science, data analysis, or writing. However, the use of this tool also raises many ethical questions among users, especially in the educational sector, where some activities such as writing essays may be disrupted.

The use of language models (LMs) in sociological research is described in (Jensen et al., 2022). The authors show that LMs are well-suited for tasks where researchers are interested in a specific minority group and cannot obtain a large amount of training data. Their illustrative analysis contributes to the sociological study of religion. They demonstrate that the proposed method is a cheap and unobtrusive way to measure religiosity in a large population in a large geographic area. Although data annotation can be very expensive, the described method is very affordable, requiring only a few thousand annotations to classify datasets of up to 10 million observations. Of note is the high versatility of the described method. The approach is applicable in any environment where researchers want to unobtrusively measure attitudes and beliefs, but surveys can be very expensive and high-quality data are scarce.

Our goal is to analyze information using tools such as LLMs, similar to the studies mentioned above. However, we want to take a different approach than simply using these tools for a given text analysis task. Our contribution is the proposal of a method that describes how to perform these analyses using efficient algorithms with much lower computational resource requirements and overall operating costs.

3 OUR PIPELINE

Our goal is to track a given phenomenon described by a phrase in a text corpus. To generalize the proposed procedure and describe it in a clear and concise way, we decided to design it in the form of a pipeline. This allows us to fully control the data flow between the individual pipeline components. Due to the reusability of individual functional blocks, we decided to divide the entire pipeline into three separate parts. We will thus have a separate part for creating the dataset,

a separate part for training the model, and a separate part for inference.

3.1 Pipeline for Creating a Dataset

1. The user selects keywords
2. We will select words that are semantically similar to the selected keywords from the corpus
3. The resulting set of words is used to select sentences from the corpus in which a word from the set occurs.
4. We will compute embeddings for the selected sentences
5. We will apply the FacilityLocation algorithm to obtain a representative sample of data
6. We will visualize this representative sample of data using the Atlas library for further analysis of these data
7. We will create a dataset from the selected sentences using LabelStudio, Atlas, or automatically using the OpenAI API.

3.2 Pipeline for Training a Model

1. We will fine-tune the Flan-T5 model on the dataset created by the previous pipeline
2. We will perform model distillation of the trained Flan-T5 model to a smaller model

3.3 Pipeline for Tracking a Given Phenomenon in a Corpus

1. We need to select all sentences that contain a keyword or a word with a similar meaning
2. We will perform the classification of the selected sentences using a distilled (small) model
3. We will compare the confidence scores. If the confidence score for a given sentence is low, we will perform the classification using the large Flan-T5 model.

3.4 Creating a Dataset

We need to create a dataset that will be used to train a model, which will then be used to track a phenomenon of our choice in the corpus. This requires us to first select all potentially relevant sentences from the corpus, which can be easily done by identifying words that describe the phenomenon and then selecting sentences that contain those words. This is the

most crude method of filtering the data and may omit relevant sentences. Nevertheless, the user may include as many words as he wants and, by this, control the recall. We expect the user to be able to determine which keywords are of interest to them. Because the corpus may contain similar words with the same meaning, we first want to determine a complete set of words on the basis of which we will filter sentences. We will determine the complete set of words using semantic similarity with the use of vector embeddings.

3.4.1 World Selection

First, we identify the individual words that describe the phenomenon to be tracked. The number of these words may vary depending on the context of the phenomenon, for example, from 5 to 20. Because there are often words that are similar or have the same meaning, we want to find such words in the corpus and take them into account. To work effectively with words from a semantic perspective, it is necessary to choose a suitable representation of these words. A commonly used approach is to convert words into their vector representation. These vectors are designed to capture hidden information about the language or analogies between words and the semantic meaning of these words. The FastText library has pre-trained models for 157 languages (Grave et al., 2018). These models are freely available online, so they are suitable for our use. We will extract only unique words from our corpus and convert them to their vector representation using the FastText library.

In this step, we obtain a vector representation of all unique words from the given corpus, as well as a vector representation of the words we selected. To select words from the corpus that are similar to the words we selected, we will use these vectors and calculate the cosine distance between the selected words and the words from the corpus. In this way, we will select the most similar semantic words n to each of the selected words using the cosine distance. In our case $n = 10$. This can be done inexpensively using libraries, such as HNSWLIB. HNSWLIB is based on the Hierarchical Navigable Small World (HNSW) algorithm (Malkov and Yashunin, 2018), which facilitates a fast approximate search for nearest neighbors. By creating a multilayer structure of navigable small-world graphs, HNSWLIB can quickly traverse data and identify the nearest data points, which is especially useful for applications for large datasets where speed and accuracy are critical.

The uppermost layers contain sparse data, allowing rapid navigation over large distances, while the lower layers become increasingly dense, allowing for finer navigation. When a search query is initiated,

the algorithm begins from the top layer and traverses down, focusing on the approximate nearest neighbors in an efficient manner. With the help of HNSWLIB, we can retrieve semantically relevant words in a few milliseconds.

3.4.2 Selection of Training Sentence

The purpose of this step is to filter out sentences that are not relevant to our target task. We have a set of selected keywords, and the fastest solution is to select only those sentences that contain at least one of the selected keywords. Due to such a rough filtering, which is also very fast, we can only work with relevant data in the next steps, which significantly affects the speed of the entire pipeline.

From the perspective of our main task, which is to calculate the frequency of occurrence of a given phenomenon in individual months, we are currently able to go through selected sentences that contain selected words, but we are not able to accurately determine whether these sentences refer to our phenomenon. This is because a single word can appear in different contexts and thus change the meaning of the sentence. Therefore, we would like to further identify only those sentences that semantically correspond to this phenomenon. A possible solution is to use a classifier. We want to train this classifier to accurately identify our phenomenon, thus further removing irrelevant sentences that contain the selected words but are in a different context. In the following steps, we want to fully focus on assembling the dataset and training the model.

As can be seen in Figure 1, the embeddings form clusters based on their semantic similarity. We would like to create a training dataset from sentences that well represent these clusters. Therefore, we select representative sentences in such a way that their neighborhoods cover most of the embedded sentences. This can be formalized as a facility location problem that can be approximately solved with algorithms for submodular optimization.

Submodular optimization is a mathematical framework for finding the best subset of a set of elements, such as data points, features, or tasks. Algorithms for submodular optimization can be accessed from Python libraries such as Apricot (Schreiber et al., 2020). Apricot provides a variety of algorithms for submodular optimization, including greedy algorithms, local search algorithms, and approximation algorithms. It also includes a number of features to visualize and evaluate the results of submodular optimization.

Facility location functions (Krause and Golovin, 2014) are a category of submodular functions that,

when maximized, select data points that aptly represent the entire data space. The principle behind the facility location function is the optimization of pairwise similarities between the data points and their closest selected point. It is imperative for the similarity metric, although user-specified, to remain non-negative; a higher value suggests increased similarity. Optimizing a facility location function can be seen as a more greedy variant of the k-medoids algorithm. After the initial data points are selected, subsequent selections tend to be in the cluster centers. This function, similar to many graph-based functions, operates on a pairwise similarity matrix. It progressively selects points that are more similar to those points whose most similar counterparts remain quite dissimilar. In other words, the data points chosen in subsequent stages tend to represent the less represented sections of the data. The mathematical representation of a facility location function is:

$$f(X, Y) = \sum_{y \in Y} \max_{x \in X} \phi(x, y) \quad (1)$$

In the equation 1, f represents the function, X denotes a subset, Y signifies the ground set, and ϕ embodies the similarity measure between two points.

In our case, the ground set Y is the set of all sentences and the subset X is the set of sentences that we want to use for the training dataset. Therefore, we want to minimize the function $f(X, Y)$ over different choices of X . The similarity of sentences ϕ can be measured using the cosine distance between the embeddings of sentences. To create the embeddings, we use the pre-trained model all-mpnet-base-v2 (Song et al., 2020) from the Sentence Transformers library (Reimers and Gurevych, 2019).

3.4.3 Data Labeling

Once we have a representative sample of sentences, we need to label them so we can use the sentences to fine-tune a pre-trained classifier. In Section 4, we will demonstrate our pipeline on two examples used to check whether it produces expected results. In one of these examples, we track the occurrence of sentences that express an opinion about the current weather. Therefore, we need the classifier to distinguish whether a given sentence/comment is about the weather or not. The proposed labels can thus have values of Weather or Irrelevant. For a more detailed division, we can choose labels such as Hot, Cold, Irrelevant, thereby teaching the model to also distinguish whether a given sentence/comment implies that the temperature is high/low. We used three different ways to label the sentences. Given the available technologies, we would like to give users the maximum

possible control over the assignment of labels to individual sentences, but at the same time, we want to focus on an appropriate level of automation. Therefore, it seems appropriate to allow for the use of multiple approaches. The most expensive way to label is to manually assign a label to each sentence based on the judgment of an expert user. This can be supplemented by other approaches that allow for partial automation, such as finding clusters and labeling whole clusters instead of individual examples. Finally, we have the most automatic way to assign labels to sentences, for example using the OpenAI API.

Manual Labeling. This kind of labeling gives the user maximum control over the quality of the labels. There is a wide range of software available for manual labeling, and some of them allow the use of pre-trained models, which can significantly simplify the task. In our case, we used LabelStudio, in which we uploaded a pre-prepared dataset containing 250 selected sentences and assigned the appropriate class to each sentence after reading it.

Labeling with OpenAI API. To maximize the automation of labeling, one can also use one of the existing commercial LLMs. In our case, we used the OpenAI API to label another 200 selected sentences.

Labeling with Atlas. As a result of the previous step described in Section 3.4.2, we have access to sentence embeddings and their corresponding sentences. We would like to analyze these data and potentially perform topic modeling, clustering, and semantic search on them. Individual sentences that form clusters have high semantic similarity. This information can be used for dataset construction, as it will facilitate labeling and allow us to easily label all sentences from a single cluster with a single label. In figure 1, you can see the individual embedding clusters that were obtained as part of the experiment in Section 4.1.

This visualization was generated using the NomicAI Atlas tool, which is used for the analysis and visualization of multidimensional vector representations in a two-dimensional space and provides tools suitable for the creation of datasets.

3.5 Training a Model

The model is the core of our entire pipeline, so it is crucial to choose the right architecture that meets the requirements of the tasks the model will be processing. It is also essential to assemble a suitable dataset on which the model will be trained.

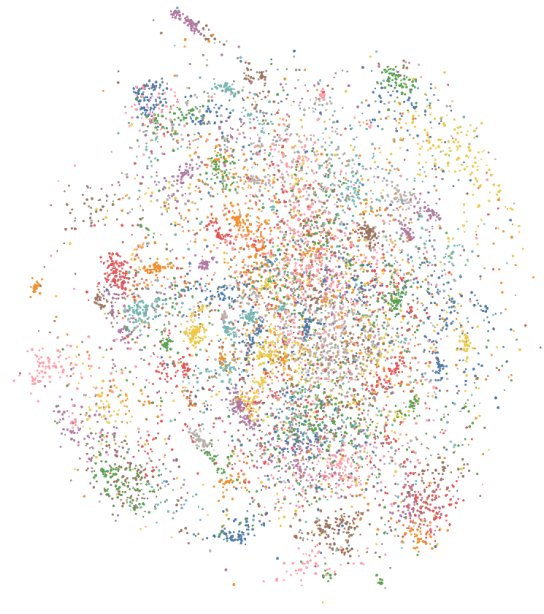


Figure 1: The sentence clusters utilized in the experiment 4.1.

The Flan-T5 model was published in (Chung et al., 2022). It is an instruction-finetuned version of the T5 model (Raffel et al., 2020), a popular LLM encoder-decoder model. This extension was created by applying the Flan finetuning procedure (Finetuning language models (Wei et al., 2021)), which means that the model is trained on a mixture of tasks, including text generation, translation, summarization, and question answering. As a result, it is able to perform a variety of NLP tasks without explicit prior training on the given task or is able to perform tasks with a few examples by in-context learning.

3.5.1 Low-Rank Adaption

Currently, the most common approach to training Large Language Models (LLMs) is to pre-train them on a general data domain and then adapt the model to a specific domain or task. With the rise of large LLMs, it is very computationally expensive to perform full fine-tuning of all the network parameters.

A technique called Low-Rank Adaptation (LoRA) (Hu et al., 2021) allows us to freeze the pre-trained weights of the model and then inject trainable rank decomposition matrices into each layer of the Transformer architecture. This allows us to very effectively minimize the number of trainable parameters for a given task.

Advantages:

- The pre-trained model can be shared and used to assemble many other small LoRA modules

for different tasks. We can freeze the shared model and effectively switch between different tasks simply by replacing the A and B matrices. This also has a significant effect on the utilization of computational resources and disk space.

- The training itself is also much more efficient, because LoRA does not need to compute gradients or maintain the state of the optimizer for a large number of parameters. Instead, it only optimizes the injected low-rank matrices, which are much smaller.

A Flan-T5 XXL model with 11B parameters was selected for training. To train with the LoRA method, we used the Peft library (Mangrulkar et al., 2022). Due to the size of the model, it was necessary to use the ability to split the model across multiple GPUs using the Accelerate library (Sylvain Gugger, 2022).

For each experiment, the model was trained with the prompts described in Section 3.5.2. The model distillation process is then described in Section 3.5.3.

3.5.2 Prompt Construction

The way we communicate with large language models (LLMs) is by providing a specific instruction based on our requirements for the output. This instruction is called a prompt. By creating a good prompt, we can guide the LLM to generate content that meets our specific requirements, reduce training costs, improve productivity, and ensure good performance. In other words, the prompt is the key to getting the most out of LLMs. By taking the time to write a clear and concise prompt, we can help the LLM to understand what we want and generate the best possible output.

Each prompt represents an instruction (Zhang et al., 2023) consisting of three parts: the instruction, which specifies the given task in a sequence of natural language, the input text `{input_text}`, which in our case represents a given sentence or comment in the corpus, and the expected output, which can be, for example, a set of possible classes into which we can classify the sentence or comment on the input.

The above prompts 2 were used for individual experiments for the training and inference of the Flan-T5 model.

3.5.3 Model Distillation

Model distillation is a machine learning technique to transfer the knowledge learned by a large and complex model to a smaller and simpler model. This is typically done by training the smaller model to predict the outputs of the larger model. Model distillation

```
text: {input_text} Is this text about
political election?
Options:
-Yes
-No

text: {input_text}
Does the text imply a hot weather,
cold weather or is not about weather.
Options:
-Hot
-Cold
-Irrelevant
```

Figure 2: Examples of used prompts: **Top:** prompt for an experiment tracking political election discussion, **Bottom:** prompt for an experiment tracking weather discussion.

has several advantages, including reducing the size and complexity of large models, improving the performance of small models, and increasing the robustness of models to noise and adversarial attacks. However, model distillation also has some disadvantages, such as being computationally expensive to train, especially for large and complex models, and distilled models can be less robust than the original models and may not perform as well on tasks that were not explicitly considered during training. In general, model distillation is a powerful technique that can be used to improve the performance, reduce the size, and increase the robustness of machine learning models.

Methods such as parameter-efficient fine tuning (PEFT) and pattern-exploiting training (PET) achieve very good results in the training of Large Language Models (LLMs). However, they are difficult to use because they are exposed to high variability in manually created prompts and typically require billion-parameter language models to achieve high accuracy. These drawbacks are addressed by an efficient and prompt-free framework for few-shot fine-tuning of Sentence Transformers (ST), called Sentence Transformer Fine-Tuning (SetFit) (Tunstall et al., 2022). This simple framework requires no prompts for training or inference, and achieves high accuracy with an order of magnitude fewer parameters and an order of magnitude shorter training time.

As mentioned above, the technique is built on top of models from the Sentence Transformer family, so it is advisable to use a model that is supported within this framework directly. Comparison in (Reimers and Gurevych, 2019) shows that the all-mpnet-base-v2 model is the best rated in terms of quality.

3.6 Pipeline for Tracking Phenomenon

This phase requires processing the entire corpus, so we want it to be as fast as possible. It is advisable to process the corpus at several levels of granularity. The first step is to select only relevant sentences from the entire corpus. This process is identical to the selection of sentences from the first pipeline. This will significantly reduce the number of sentences for which we will perform inference. We then classify the selected sentences as the second step using a distilled model.

The third step is to correct for those predictions for which the distilled model is not confident. Here, it is crucial to choose a threshold in a suitable way with which we will compare the confidence that the model returns for each predicted sentence. For each task, it is necessary to analyze all the predictions and determine the threshold value based on this analysis. Thanks to the appropriately chosen threshold value, we will only perform predictions with a large model for those sentences for which it is absolutely necessary, and we assume that there will be few of these sentences.

4 EXPERIMENTS

To demonstrate the results of our approach, we performed two experiments. We selected experiments that follow discussions about weather and political elections because the results of these experiments are easily verifiable. Both experiments were built on the same pipeline, with the same models, and on the same dataset. The dataset contains sentences from social media collected from 2019 to 2022. The sentences are also divided into individual months in each year. On average, there are 4.5 million sentences available per month. The entire dataset for all years contains over 200 million sentences. The specific number of sentences for individual years can be found in Table 1.

Table 1: Dataset sentences per year.

Year	Sentences
2019	54442253
2020	54715529
2021	54741599
2022	54456946

4.1 Political Elections

A use case for this experiment is that a user wants to find out how much people comment about political elections on social media in different months. This

is the simplest version of the experiment, where we only look for sentences that are related to political elections. On the other hand, we want to exclude all sentences that are not related to political elections. In essence, we want to verify the functionality of our proposed approach in this experiment. For this experiment, we have selected the keywords "election" and "vote", which we expect to be the most common words and semantically similar words in political comments.

Table 2: FastText based filtering for political election experiment.

Year	2019	2020	2021	2022
Time	15.72s	15.24s	18.0s	16.44s
Sentences	42332	58949	34493	32412

Table 3: SetFit based filtering performance.

Year	2019	2020	2021	2022
Time	50.80s	68.75s	43.51s	38.93s

You can find information about the performance of the individual pipeline components in table 2, which provides information about how many sentences the FastText algorithm selected based on the given requirements and how long the filtering took. You can see the times for classifying sentences using the SetFit model in Table 3.

4.2 Discussion About Weather

Table 4: FastText based filtering.

Year	2019	2020	2021	2022
Time	79.8s	82.56s	83.28s	89.28s
Sentences	161747	152994	157420	161617

Table 5: SetFit based filtering performance.

Year	2019	2020	2021	2022
Time	219.54s	212.42s	215.45s	216.41s

In the second experiment, we would like to analyze user discussions about the weather in relation to the individual months of the year. We expect that the discussion will change over the course of the year and that in hot months, discussion about hot weather will be more frequent than discussion about cold weather, while in cold months, discussion about cold weather will be more frequent than discussion about hot weather. For this experiment, we selected the following keywords: heatwave, sunny, wildfire, muggy, wet, windy, tornado, cloudy, rain, rainy, weather, beach.

5 RESULTS

We verified the above experiments on the dataset described in 4. In practice, this involves determining the frequency of occurrence of the observed phenomenon in a corpus of text divided into 12 months for the years 2019 to 2022. For the tracking of political elections, you can see the frequency of occurrence of such a discussion in Figure 3, 4, 5, 6. Since the original data source is the social network Reddit and the majority of the discussants come from the United States, we can infer the following phenomena:

- **The 2020 United States Presidential Election.** Joe Biden defeated Donald Trump and became the 46th president of the United States.
- **The 2022 United States House of Representatives Elections.** Republicans won a majority in the House of Representatives, ending the Democrats’ two-year majority
- **The 2022 United States Senate Elections.** Democrats maintained a narrow majority in the Senate.

Specific events may be relevant for specific months:

- **May 2020.** In this month, the 2020 United States presidential primary elections took place.
- **September 2020.** In this month, the presidential debates between Joe Biden and Donald Trump took place.
- **November 2020.** The 2020 United States presidential election took place in this month.
- **February 2022.** The 2022 United States House of Representatives elections took place in this month.
- **April 2022.** In this month, the 2022 United States Senate primary elections took place.
- **November 2022.** The 2022 United States Senate elections took place in this month.

The results of the second experiment, which are described in 4.2, can be seen in figure 7, 8, 9, 10. This experiment was about people’s discussions of hot summer weather and cold winter weather. The graphs show that the frequency of these discussions corresponds to the seasons of the year. In the winter, people discuss cold weather more often, and in the summer, they discuss hot weather more often.

Next, we summarize our experiments in terms of performance, which we believe is the biggest benefit of the proposed approach. All experiments were conducted on an NVIDIA DGX-1 machine. Due to the size of the Flan-T5 model in the XXL version,

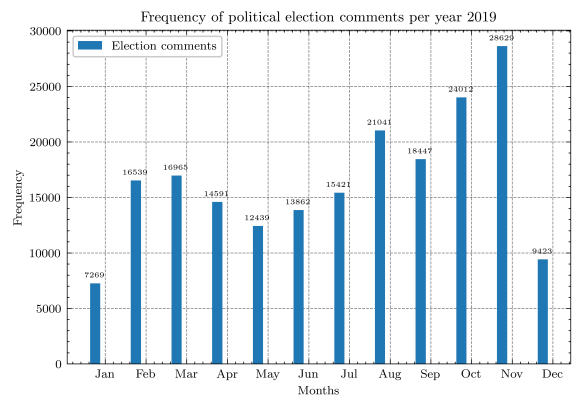


Figure 3: Results for political election discussion, year 2019.

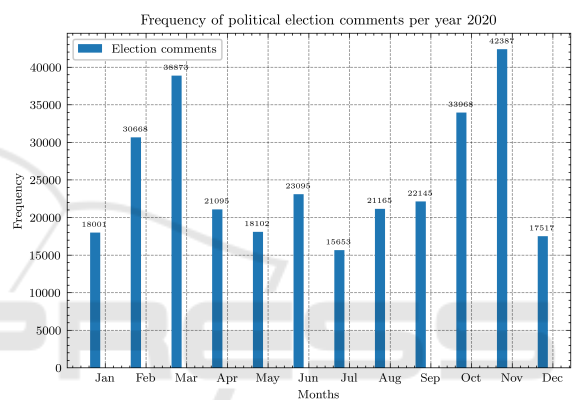


Figure 4: Results for political election discussion, year 2020.

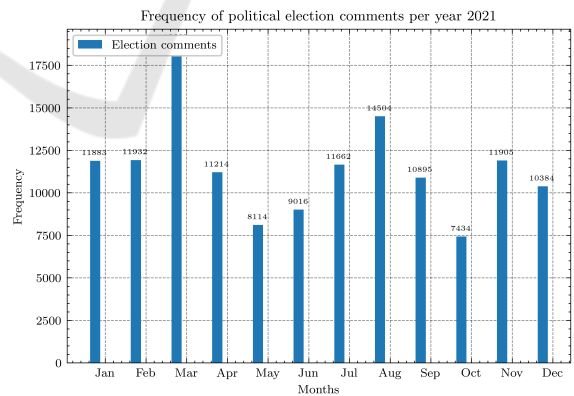


Figure 5: Results for political election discussion, year 2021.

which uses 11B parameters, all 8 graphics cards were used. One graphics card was used to work with the SetFit model. First, we would like to emphasize that the dataset used, described in section 4, contains a total of 218,356,327 sentences. For the prediction of the Flan-T5 model, we achieved an average time of

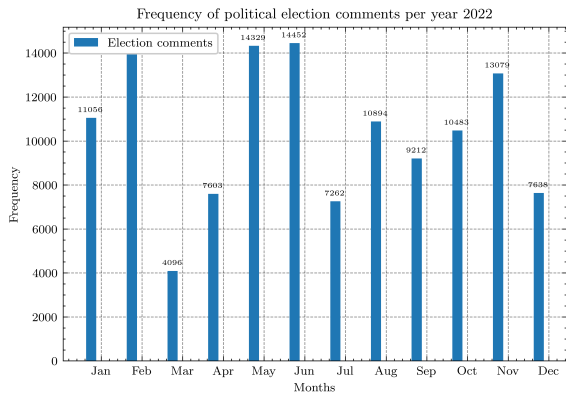


Figure 6: Results for political election discussion, year 2022.

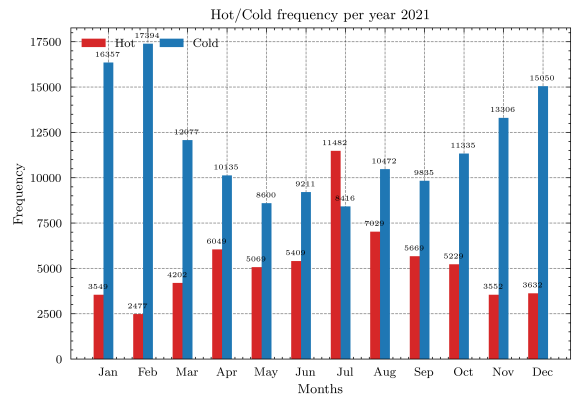


Figure 9: Results for discussion about the weather, year 2021.

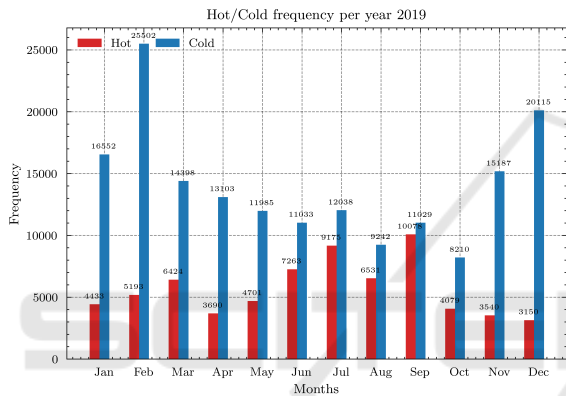


Figure 7: Results for discussion about the weather, year 2019.

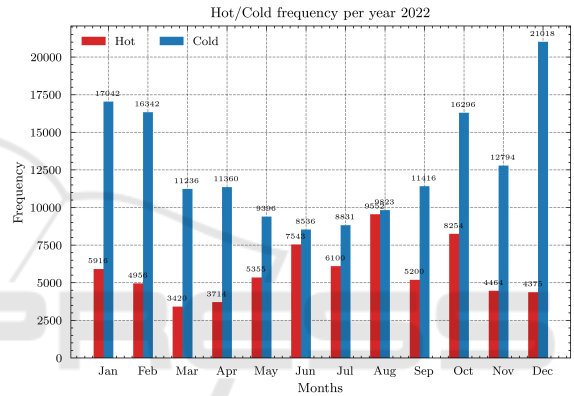


Figure 10: Results for discussion about the weather, year 2022.

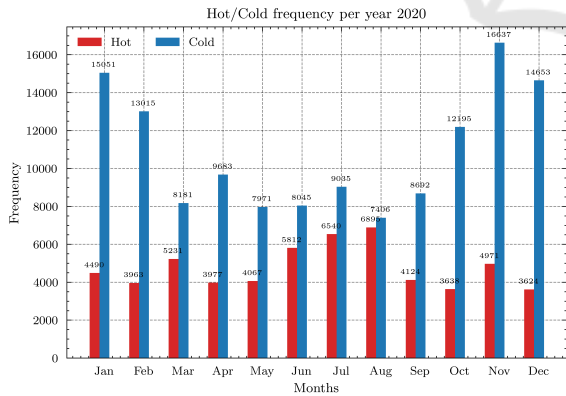


Figure 8: Results for discussion about the weather, year 2020.

0.2052 seconds, and for the prediction of the SetFit model, we achieved an average time of 0.0011 seconds. If we were to analyze the entire dataset using only the Flan-T5 model, we would reach a total time of 12,446 hours, and with the SetFit model, a total

time of 67 hours. Here, you can see the strength of the proposed solution, when we first select the desired sentences, which are orders of magnitude fewer, using a very fast filtering process. We then apply the prediction to this set using the SetFit model, which is very fast. For sentences where the confidence of the SetFit model is less than 0.98, we refine them using the slower Flan-T5 model. Thanks to this, the overall analysis can be very fast as you can see in table 6.

Table 6: Average filtering performance in minutes.

Experiment	FastText	SetFit	Flan-T5
Political Election	1.09m	3.08m	230.07m
Weather	5.58m	11.61m	671.93m

6 CONCLUSIONS

In this contribution, we presented a novel pipeline design for the analysis of extensive corpora, with an emphasis on tracking specific phenomena. Our find-

ings underscore the pivotal role of data preprocessing, highlighting its impact on subsequent stages, especially when interfaced with LLMs. Our incremental approach offers several advantages, including flexibility and the ability to adapt individual steps to align with specific user goals. This ensures that users can effectively sift through vast information sources, differentiating between pertinent and non-pertinent data, ultimately facilitating a more concentrated investigation of the investigated phenomenon.

However, we also acknowledge certain limitations inherent in our approach. A significant challenge is the preparatory phase of data curation and model training, which we aim to alleviate in future iterations through enhanced automation of the pipeline. Additionally, the Flan-T5 model's intensive hardware demands underscore the time-intensive nature of analyzing sizable corpora.

In spite of these challenges, we believe that our approach has several advantages. Due to the way the pipeline is designed, users are allowed to customize the solution. Ultimately, users can decide for themselves how accurate they need the results at each step and the associated computational cost of each step. Thanks to this flexibility, it is possible for users to optimize the task according to their requirements, either for computational cost and speed or for overall calculation accuracy.

ACKNOWLEDGEMENTS

The work is partially supported by grant SGS13/PřFMF/2023. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254).

REFERENCES

- Biswas, S. S. (2023a). Potential use of chat gpt in global warming. *Annals of biomedical engineering*, 51(6):1126–1127.
- Biswas, S. S. (2023b). Role of chat gpt in public health. *Annals of biomedical engineering*, 51(5):868–869.
- Bukar, U., Sayeed, M. S., Razak, S. F. A., Yogarayan, S., and Amodu, O. A. (2023). Text analysis of chatgpt as a tool for academic progress or exploitation. *Available at SSRN 4381394*.
- Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al. (2022). Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.
- Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.
- Jensen, J. L., Karell, D., Tanigawa-Lau, C., Habash, N., Oudah, M., and Fairus Shofia Fani, D. (2022). Language models in sociological research: an application to classifying large administrative data and measuring religiosity. *Sociological Methodology*, 52(1):30–52.
- Krause, A. and Golovin, D. (2014). Submodular function maximization. *Tractability*, 3(71-104):3.
- Krippendorff, K. (2018). *Content analysis: An introduction to its methodology*. Sage publications.
- Macanovic, A. (2022). Text mining for social science—the state and the future of computational text analysis in sociology. *Social Science Research*, 108:102784.
- Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.
- Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., and Paul, S. (2022). Peft: State-of-the-art parameter-efficient fine-tuning methods. <https://github.com/huggingface/peft>.
- Patel, S. B. and Lam, K. (2023). Chatgpt: the future of discharge summaries? *The Lancet Digital Health*, 5(3):e107–e108.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.
- Reimers, N. and Gurevych, I. (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.
- Schreiber, J., Bilmes, J., and Noble, W. S. (2020). apricot: Submodular selection for data summarization in python. *Journal of Machine Learning Research*, 21(161):1–6.
- Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. (2020). MpNet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*.
- Spörlein, C. and Schlueter, E. (2021). Ethnic insults in youtube comments: Social contagion and selection effects during the german “refugee crisis”. *European Sociological Review*, 37(3):411–428.
- Sylvain Gugger, Lysandre Debut, T. W. P. S. Z. M. S. M. M. S. B. B. (2022). Accelerate: Training and inference at scale made simple, efficient and adaptable. <https://github.com/huggingface/accelerate>.
- Törnberg, A. and Törnberg, P. (2016). Combining cda and topic modeling: Analyzing discursive connections between islamophobia and anti-feminism on an online forum. *Discourse & Society*, 27(4):401–422.

- Törnberg, P. (2023). How to use llms for text analysis. *arXiv preprint arXiv:2307.13106*.
- Tunstall, L., Reimers, N., Jo, U. E. S., Bates, L., Korat, D., Wasserblat, M., and Pereg, O. (2022). Efficient few-shot learning without prompts. *arXiv preprint arXiv:2209.11055*.
- Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Fine-tuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.
- Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., et al. (2023). Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

