ITC6110_NLP_Report_Kapsalis_Stavrogiannis_Tzoras_Final.docx

by Konstantinos Tzoras

Submission date: 28-Jul-2024 08:19PM (UTC+0300)
Submission ID: 2423685230
File name: ITC6110_NLP_Report_Kapsalis_Stavrogiannis_Tzoras_Final.docx (5.12M)
Word count: 11748
Character count: 68412

ITC6110 - Natural Language Processing

Final Project Report

Instructor: Dr. E. Chatzimichali

Submitted by:
Kapsalis C.
Stavrogiannis C.
Tzoras K.

53

Table of Contents

TABLES

Table of Figures

ABBREVIATIONS

AI - Artificial Intelligence
ML - Machine Learning
DL - Deep Learning
MT - Machine Translation
CPU - Central Processing Unit
GPU - Graphics Processing Unit
URL - Uniform Resource Locator
NMT - Neural Machine Translation
NER - Named Entity Recognition
MLP - Multi-Layer Perceptron
RegEx - Regular Expressions
SOTA - State Of The Art
GRU - Gated Recurrent Units
LSTM - Long Short-Term Memory
BLEU - Bi-Lingual Evaluation Understudy
Seq2Seq - Sequence-to-Sequence
ME - Maximum Entropy

1. Introduction

This project demonstrates practical applications of key techniques in Natural Language Processing (NLP). It encompasses all stages of the NLP pipeline, from data collection and preprocessing to feature engineering, model development, and testing. We have gathered a suitable dataset for supervised learning tasks, clean and normalize the text data, and convert it into numerical representations using techniques such as Bag of Words, TF-IDF, or word embeddings. Visualizations helped detect patterns and clusters within the data. The project includes building models for both unsupervised learning, topic modeling, and supervised learning, including text classification and Named Entity Recognition (NER). We will also explore additional tasks, machine translation and text summarization, to further demonstrate the model's capabilities. Throughout, we ensured rigorous model training, testing, and refinement using appropriate performance metrics, culminating in a detailed report and presentation of our methodologies and findings.

2. Data Collection and Loading

2.1. Overview

We proceeded to the selection of a group of publicly available Kaggle datasets that allow for the smooth implementation of the various Natural Language Processing (NLP) techniques we were taught in the Natural Language Processing course during the Spring Semester, 2024.

2.2. Methodology

The various datasets had a focus on a different task. Here we present their titles, source, and corresponding dataset IDs we assigned to them.

Table 1: Unsupervised Learning-Focused Datasets

Dataset Title	Dataset URL	Dataset ID
Coronavirus (covid19) Tweets - early April	https://www.kaggle.com/datasets/smid80/coronavirus-covid19-tweets-early-april	• 'uns_covid1' (un-processed), • 'uns_prep_covid1' (after pre-processing)
Twitter BTC	https://www.kaggle.com/datasets/tylerdurden73/twitter-btc	'uns_crypto_btc1'
Customer Support on Twitter - Customer Support on Twitter	https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter	• 'uns_cust_sup_cs1' (un-processed), • 'uns_prep_cust_sup_cs1' (after pre-processing)

Table 2: Text Classification-Focused Datasets

Dataset Title	Dataset URL	Dataset ID
Hate Speech and Offensive Language Detection	https://www.kaggle.com/datasets/thedevastator/hate-speech-and-offensive-language-detection	• 'c_hate_speech1' (un-processed), • 'c_prep_hate_speech1' (after pre-processing)
Sentiment140 dataset with 1.6 million tweets	https://www.kaggle.com/datasets/kazanova/sentiment140	'c_sentiment'
Twitter Information Operations Classification	https://www.kaggle.com/datasets/pookiewiggington/twitter-information-operations-classification	'c-bottling'
Emotions	https://www.kaggle.com/datasets/nelgiriyewithana/emotions	• 'c_emotions1' (un_processed), • 'c_prep_emotions1' (pre-processed)
MBTI Personality Type Twitter Dataset	https://www.kaggle.com/datasets/mazlumi/mbti-personality-type-twitter-dataset	• 'c_MBTI_personality' (un-processed), • 'c_prep_MBTI_personality' (after pre-processing)

Table 3: Machine Translation-Focused Datasets

Dataset Title	Dataset URL	Dataset ID
WMT 2014 English-German	https://www.kaggle.com/datasets/mohamedlotfy50/wmt-2014-english-german	'wmt2014'

All these datasets contained millions of text samples and we wanted to work on a big scale in order to produce high-quality results. To facilitate this, we used Google Cloud's BigQuery (n.d.) to save our raw and processed datasets as well as google drive to store other project-related files in a common data store that allowed for rapid file & data sharing and processing across our systems.

All our operations were hosted on Google's Colaboratory service (Google, n.d.), which offers extended options in computational units to use and facilitates significant interoperability functionalities for teams to utilize.

3. Text Pre-processing

3.1. Overview

Text pre-processing is the process of constructing structured data stores out of raw text in a way that minimizes randomness in the data included. The goal is to remove in-text elements that do not bear any contextual information, while ensuring consistent representations of similar utterances. These steps help the algorithms implemented converge faster and better uncover the underlying relationships in our data.

3.2. Methodology

As all datasets used contained texts originating strictly from X (formerly known as Twitter) posts, we followed the same approach across all of them. Only text samples in the English language were considered.

In the same algorithm we implemented for text pre-processing we also performed the extraction of features of linguistic interest (as in the number of Part of Speech-POS and Named Entity instances contained in each text) to be taken into account in some of the datasets used for text classification.

It is important to note here that all steps implemented -except for converting texts to the lower case- and especially lemmatization were not implemented on the data to be used with Deep Learning techniques. This is because the relevant models are very successful at capturing context, and lemmatization might hinder the extraction of information crucial to making the most accurate predictions; these models are capable of quickly and effectively recognizing (for example) how a verb is conjugated and derive the meaning of this specific setting.

For all of the following operations, we made use of Python's 're' (Python Software Foundation, n.d.), 'NLTK' (Bird et al., 2009), 'SpaCy' (Explosion, n.d.), and 'Pandas' (McKinney, 2010) modules.

3.2.1. Text Normalization

The first operation implemented on our data was checking for duplicate values and dropping them. Multiple entries with the exact same characteristics could contribute to overfitting, and if there are many of them they could also slow down training. The availability of high volumes of data allowed us to also drop rows containing missing values in any of the columns initially contained in them, as they would render them useless for exploratory analysis or inference.

The specifics of our normalization steps are detailed in the Appendix (see Appendix I).

Table 4: Example of text sample transformations for different levels of preprocessing

(raw) 'text'	**'light_cleaned_text'**	**'clean_text'**
"Welcome to the family, Mia Meli! #new #puppy #adorable #cute #fluffy #mutt #mybaby #girl http://t.co/a27C1XueZP"	"welcome to the family, mia meli!"	"welcome the family mia meli!"

3.2.2. Removal of Context-Irrelevant Information

Several steps were taken to drop utterances carefully deemed as not bearing any useful contextual information (see Appendix II).

3.3. Results

The features extracted from the raw text have as follows (based on the 'uns_covid_1' dataset which contains only two features/columns - 'text' and 'label'):

Figure 1: Features Extracted from Initially Unprocessed Datasets used in Text Classification

In the rest of datasets (used in unsupervised learning or available fully pre-processed), we only extracted the 'ELIPSIS_instances', 'upper_words', 'tweet_length', 'no_sentences', 'has_username', and 'has_hashtag' features. In those that came fully pre-processed, we performed no extra operation.

Due to the large scale of our operations (using millions of tweets at a time), our computers -and even Google's rented machines- required very long execution times. This led us to the investigation of techniques that make use of all available computational capacities of our machines and execute all underlying parts of our algorithm with the most efficient code possible.

Each occurrence of the task of pre-processing different text sample inputs is independent from each other. Thus we can perform the corresponding computations on different CPU (on, in our case, TPU) cores simultaneously to maximize the efficiency with which we utilize our computational resources.

Parallel computing and batching implementations in python helped us greatly. We achieved significant improvements in computation time, managing to fully pre-process about 2,500,000

tweets per 30 mins (for example on the 'uns_cust_sup_cs1' dataset) using Google's TPU via Google Collaboratory (Google, n.d.):

Figure 2: Execution time for the full pre-processing of the 'text' column in the 'uns_cust_sup_cs1' dataset containing about 2,500,000 text samples.

4. Topic Modeling

Overview

The primary aim of this process is to extract and understand topics within a textual dataset using various techniques. The approach involves evaluating the effectiveness of different topic modeling methods, measuring the coherence of topics, and visualizing topic distributions to gain insights into the structure and content of the data. The techniques employed include Latent Semantic Analysis (LSA), BERTopic with sentence embeddings and UMAP, and Latent Dirichlet Allocation (LDA) with t-SNE.

The same process has been applied in two distinct datasets, one associated to covid-related tweets and one with customer support comments.

Minor additional preprocessing steps were applied and the main preprocessing of both datasets will not be included here due to word limitations since it is also explained in previous chapters and follows a common approach for all datasets.

4.1. Models Applied

Latent Semantic Analysis (LSA)

Latent Semantic Analysis (LSA) is employed to identify and analyze topics from a collection of text documents. The process begins with text preprocessing, where documents are tokenized and converted into a matrix of term frequencies using CountVectorizer. This matrix captures the frequency of each term in the documents. Additionally, dimensionality reduction is applied through Singular Value Decomposition (SVD), specifically using TruncatedSVD. This step reduces the number of dimensions in the term-document matrix to a specified number of topics, allowing us to discover latent semantic structures in the data (Dasgupta, 2021). Moreover, to assess the quality of the topics, coherence scores are calculated. This is done by creating a dictionary and corpus from the tokenized documents and using gensim's CoherenceModel to compute the coherence scores. A higher coherence score indicates that the topic is more interpretable and meaningful.

Additionally, we measured how distinct the topics are from one another using the diversity metric in which topics are evaluated by calculating the ratio of unique words to the total number of words.

BERTopic with Sentence Embeddings and UMAP

BERTopic is a topic modeling approach that leverages sentence embeddings to capture semantic meanings of documents. This technique begins by encoding documents into high-dimensional vectors using the SentenceTransformer. Specifically, in our case we use the 'all-MiniLM-L6-v2' model . These embeddings capture the underline semantic information of each document, which is crucial for effective topic modeling. Once embeddings are generated, dimensionality reduction is performed using UMAP (Uniform Manifold Approximation and Projection) which is a technique that reduces the high-dimensional embeddings to a lower-dimensional space (typically 2D) for visualization purposes. Finally, BERTopic is fitted to the reduced embeddings so we can

extract topics and identify representative documents for each topic. The resulting topic distributions are visualized using UMAP.

Latent Dirichlet Allocation (LDA) with t-SNE

Latent Dirichlet Allocation (LDA) is another robust topic modeling technique that assigns topics to documents based on probabilistic distributions. The process begins by training the LDA model on the text corpus, using hyperparameters such as the number of topics, alpha (document-topic density), and eta (topic-word density). After training, the document-topic matrix is computed, which captures the distribution of topics within each document. To visualize these distributions, dimensionality reduction is applied using t-SNE (t-distributed Stochastic Neighbor Embedding). t-SNE reduces the document-topic matrix to two dimensions, creating a 2D plot that reflects the similarity and distribution of topics across documents offering visualization which allows for identification of clusters of topics and their separation.

4.2. Analysis of Topic and Clustering

Customer Support Case

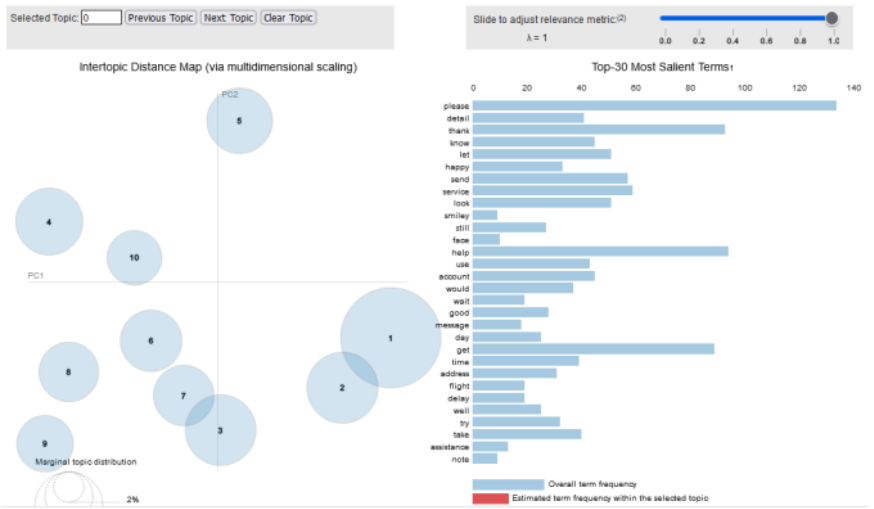

Figure 3: LDA clustering-topic visualization for the basic LDA model on customer support dataset.

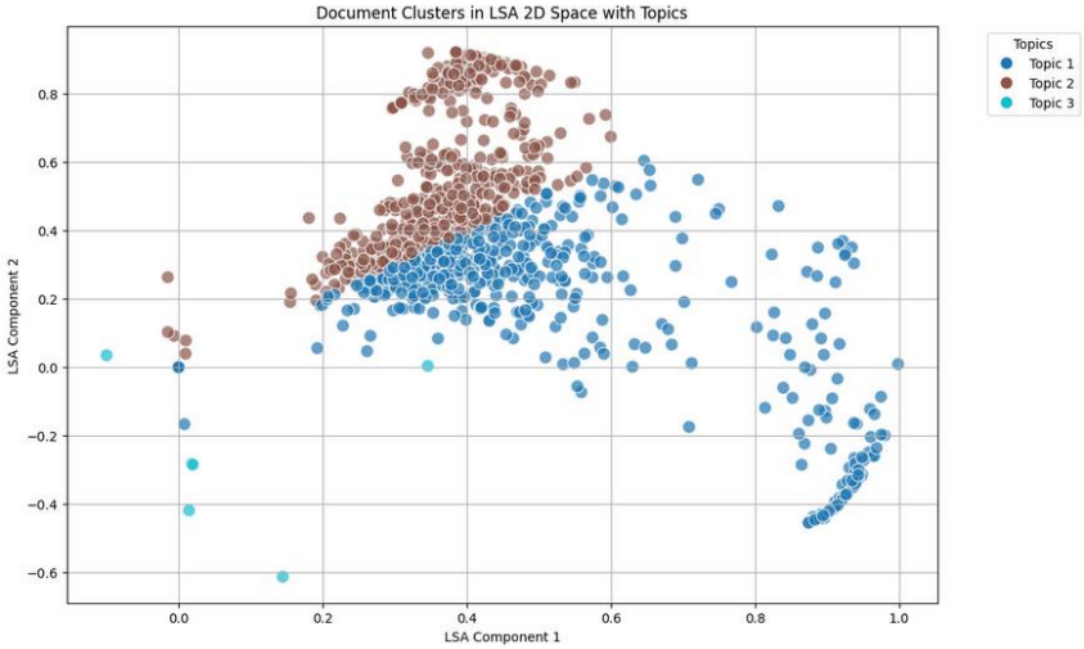

Figure 4: : LSA visualization for the customer support dataset using TruncatedSVD for dimensionality reduction (decompose the document-term matrix in topics).

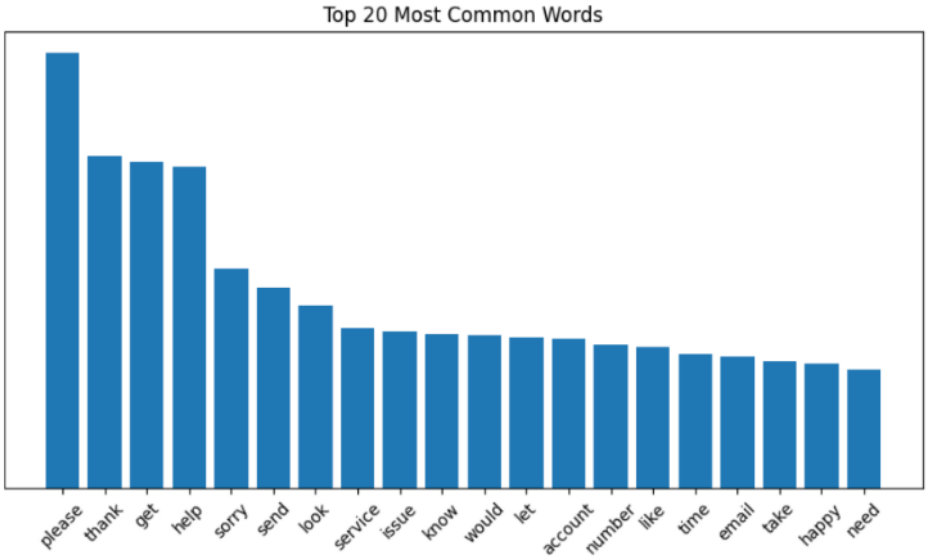

Figure 5: Top 20 most common unigrams of the customer support dataset. The first words seem to relate to the sentiment of the customers.

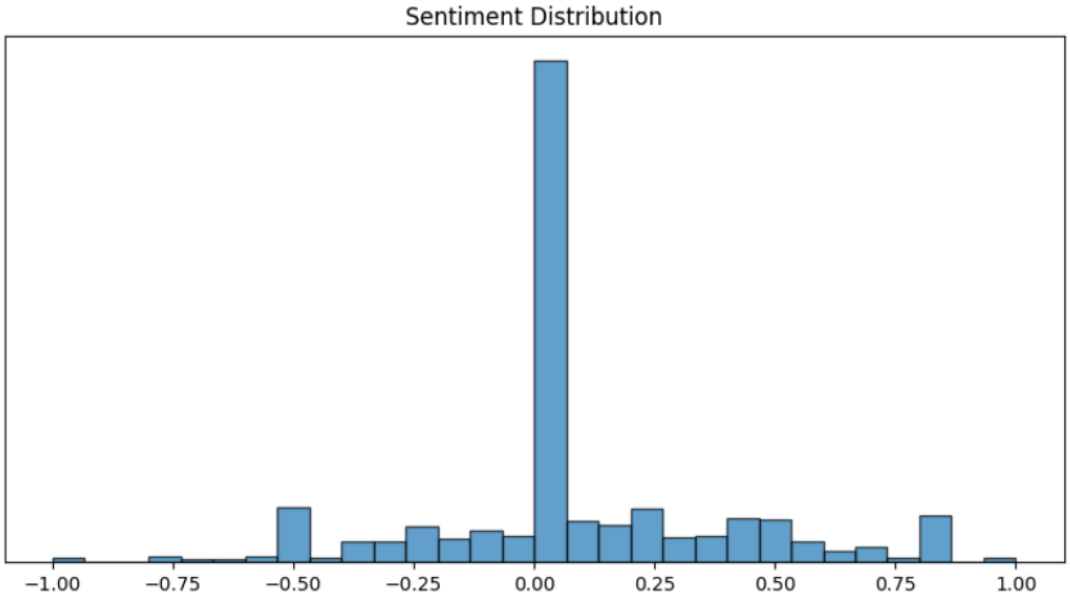

Figure 6: Sentiment analysis on customer support dataset. This seems to follow a normal distribution, which can signify that most of the comments/tweets are mainly asking the customer support service rather than expressing an opinion (negative or not on the subject).

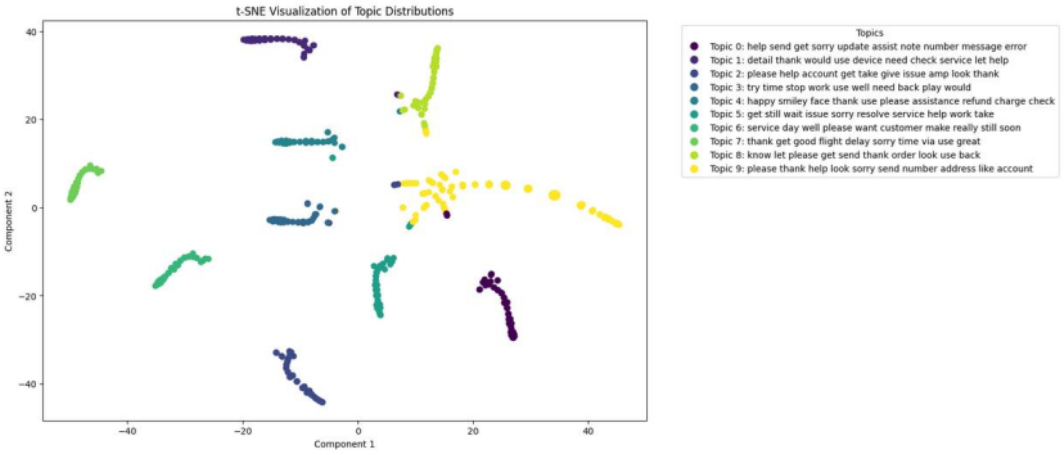

Figure 7: Basic LDA with t-SNE for dimensionality reduction on customer support dataset.

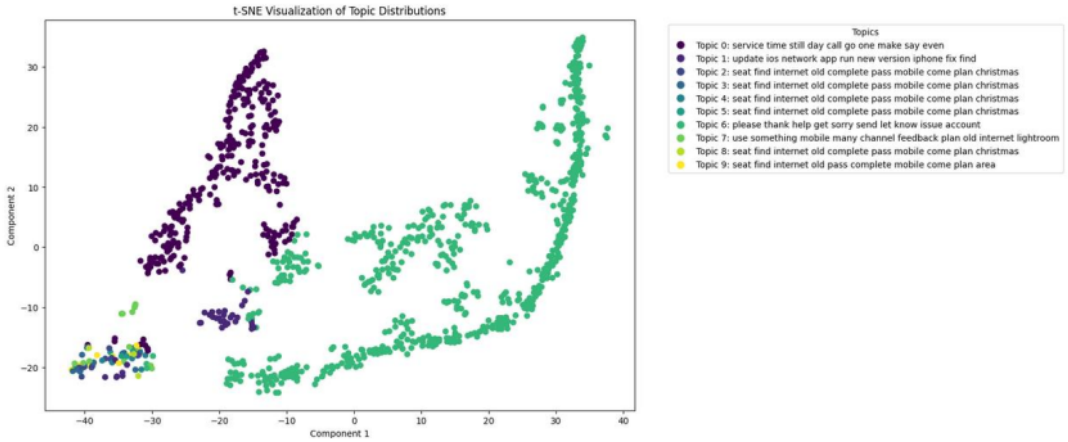

Figure 8: LDA best parameters (highest coherence score) visualization with t-SNE for dimensionality reduction on customer support dataset.

The t-SNE visualizations of topic distributions generated by two different LDA models, one with the best parameters and the other a basic model without parameterization, reveal notable differences in clustering and topic separation.

The visualization of the best parameterized LDA model (figure 1) shows less defined and more scattered clusters with significant overlap, indicating less distinct topic separation despite parameter optimization. In contrast, the basic LDA model without parameterization exhibits clearer and more distinct clusters with minimal overlap, suggesting more effective topic separation. The keywords for each topic in the best parameterized model are more specific but do not translate into clear visual clusters, whereas the basic model has fewer specific keywords but results in clearer visual distinctions. This counterintuitive outcome may be due to overfitting, t-SNE sensitivity, or the characteristics of the dataset used.

Finally, to address this, we would recommend (next steps) adjusting t-SNE parameters or even explore alternative visualization methods like UMAP. As it stands it would be correct to potentially combine insights from both models.

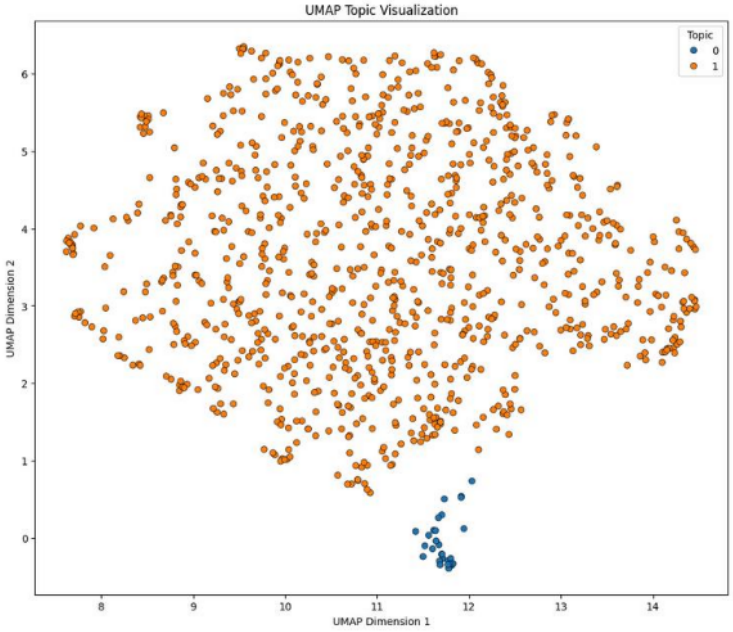

Figure 9: Topic visualization for the best parameter BERTopic model using UMAP for dimensionality reduction on the customer support dataset.

On the above figure we can see that the best parameter BERTopic does create only two topics where all the info is clustered. From this plot, one can infer that the BERTopic model has identified at least two distinct topics, with one being significantly more prominent in the dataset than the other. The tight clustering of Topic 0 suggests it represents a niche or specialized topic within the data, while Topic 1 covers a broader or more general area.

Covid Case

Results and visualizations derived from the Covid dataset.

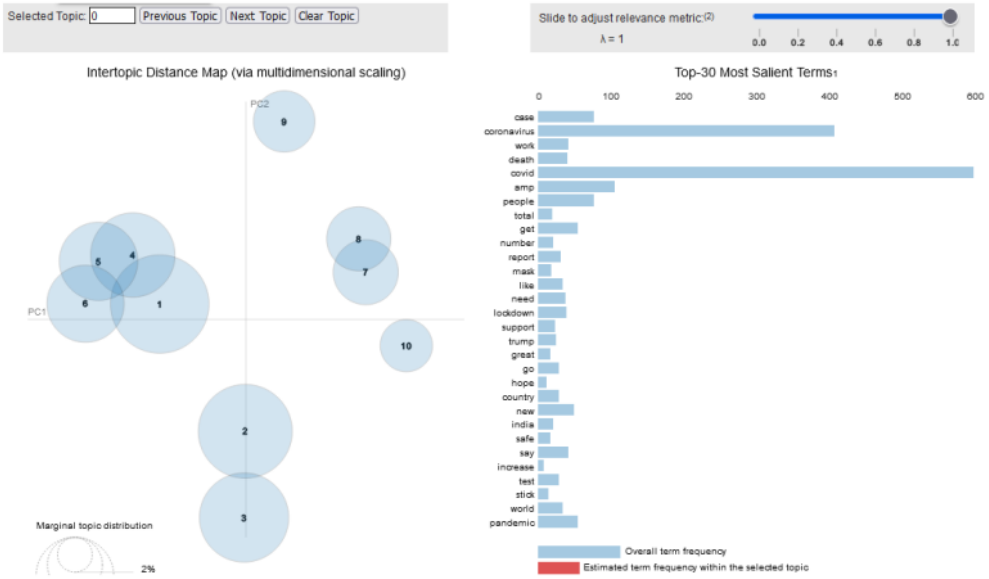

Figure 10: LDA clustering-topic visualization for the basic LDA model on covid dataset.

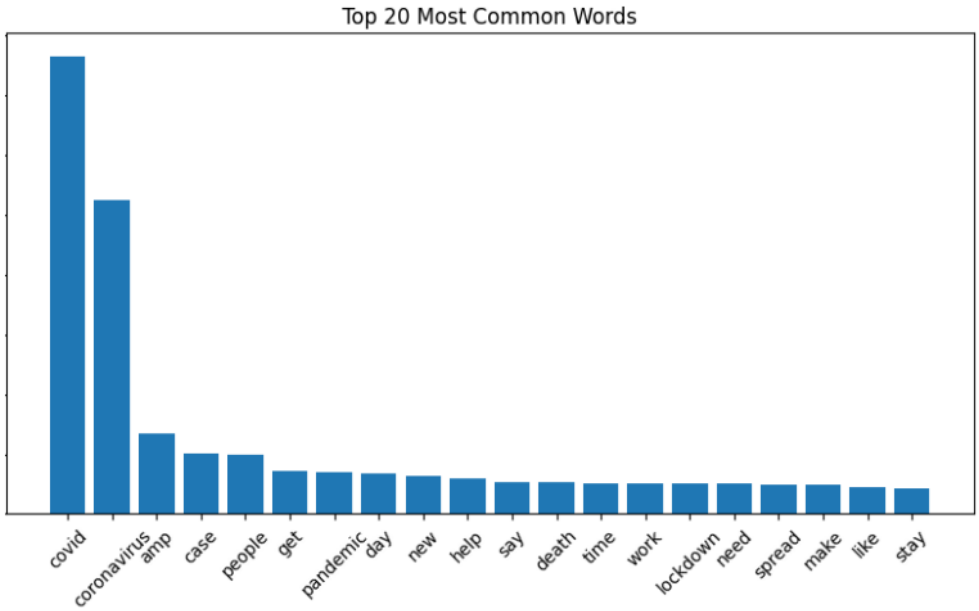

Figure 11: Most common unigrams for the covid dataset. While this that it might cause an issue regarding the commonality of the coronavirus and covid keywords, we leave it as is since it is very common to only have one word besides them in our document.

We chose not to remove these words though since in many cases we only have one word besides the covid/coronavirus keywords (taking into account that the dataset is consisted of tweets). As such we didn't want to lose the nuance meaning of our text(s). Nevertheless, this is very well handled though by the best model as it is defined below (BERTopic) which under the hood adds less weight to words that appear very commonly within our corpus by the use of the embedding which is applied with BERTopic and utilizes transformer-based models (e.g. BERT) to generate dense word embeddings. These embeddings capture the semantic meaning of words in context rather than relying solely on their frequency (Ar5iv, n.d.).

Additional information regarding the bi-grams and tri-grams examination can be found in Appendix VII.

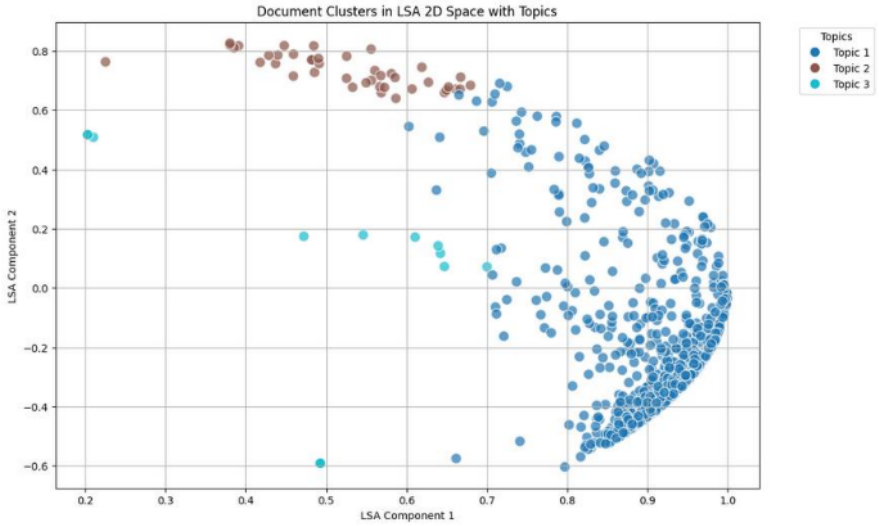

Figure 12: LSA visualization for the Covid dataset using TruncatedSVD for dimensionality reduction (decompose the document-term matrix in topics).

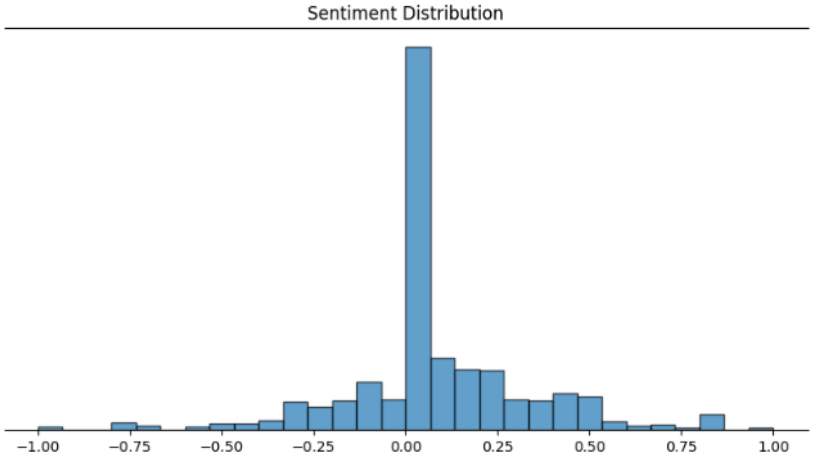

Figure 13: Sentiment analysis for Covid dataset. This seems to follow a normal distribution, which can signify that most of the comments/tweets are mainly comment on the covid-19 subject rather than expressing an opinion (negative or not) on the subject.

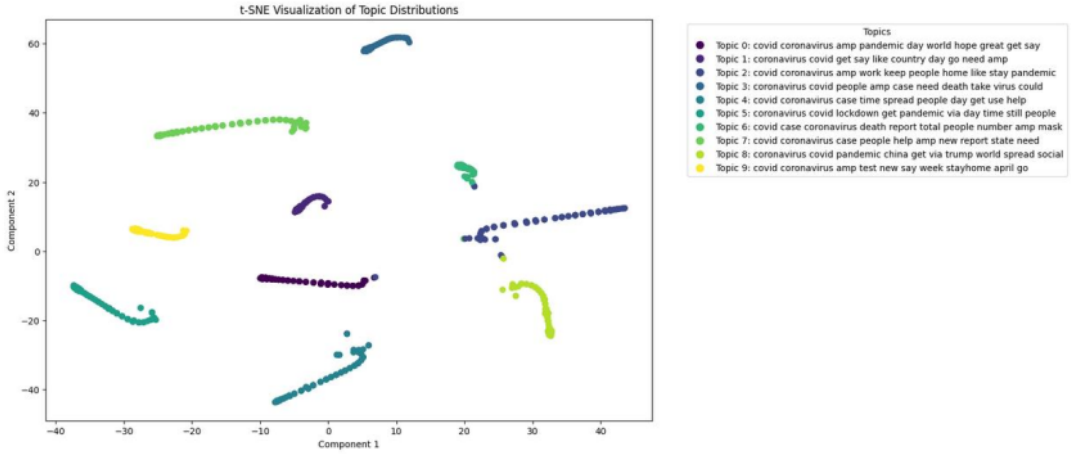

Figure 14: Basic LDA with t-SNE for dimensionality reduction on covid dataset.

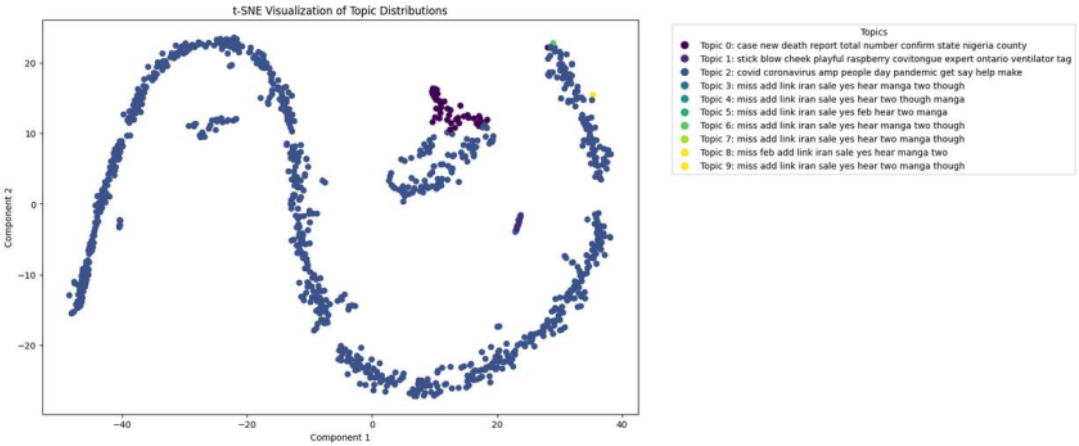

Figure 15: LDA best parameters (highest coherence score) visualization with t-SNE for dimensionality reduction on covid dataset.

From the two graphs above we can again derive that the basic model (while it has worst evaluation) provides better separation of topics and subsequently better visualization of the corpus.

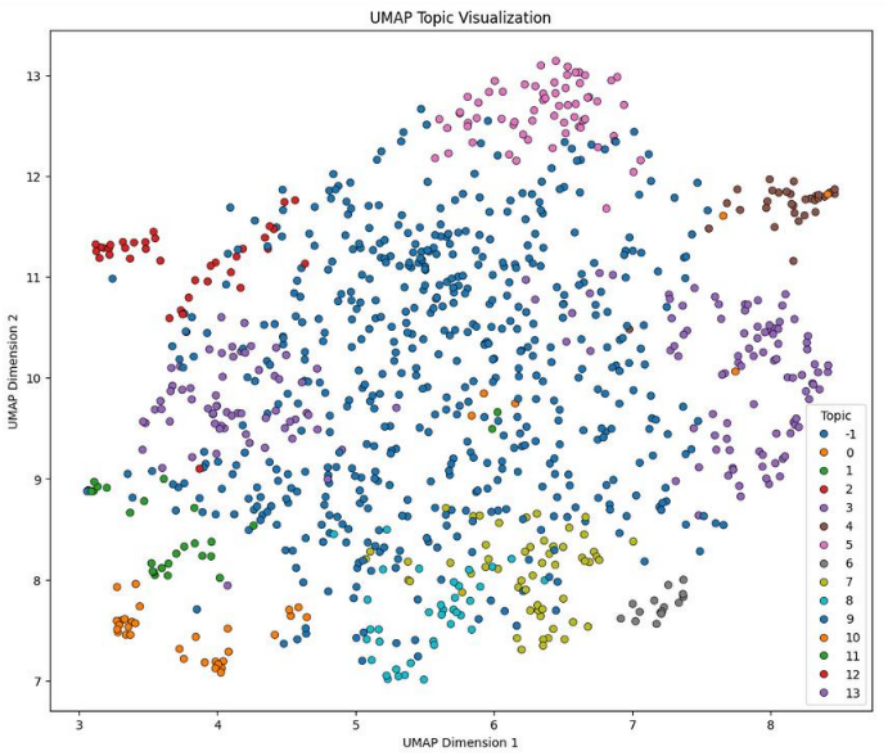

Figure 16: Topic visualization for the best parameter BERTopic model using UMAP for dimensionality reduction on the covid dataset.

We see a difference for the covid dataset compared with the same results from the customer support dataset. Here the BERTopic best model has identified several topics which are not well separated between them.
This is probably due to the fact that we have always a common theme (covid) that is clearly identified within the corpus. However, based on the embedding provided with BERTopic we see that indeed different clusters are formed , so we can say that while the basis of the documents do share similarities (and thus are not clearly separated) we do get several different topics based on the secondary/underline theme of each tweet (covid tweets is common to touch upon themes related to politics, sentiments or information, medical or otherwise, which can be identified as several different secondary themes).

Additional, EDA visualizations and details can be found in Appendix VII.

4.3. Final Evaluation and Comparison

To determine the most effective topic modeling approach, results from LSA, BERTopic, and LDA are compiled and compared in a common dataframe using a set of evaluation metrics for each. However, our approached focused on evaluating mainly the coherence scores which was the common metric that could be derived from all three (3) models tested. Moreover, we calculated the diversity, perplexity as well as silhouette score and davies-bouldin-index (LDA). Finally, we created visualizations to assess the performance of each model.
The best-performing model (BERTopic) is selected based on the above conditions. Results can be seen (for both cases) in the figures below.

	num_topics	alpha	eta	iterations	gamma_threshold	coherence	perplexity	silhouette_score	davies_bouldin_index	lda_model	nr_topics	min_topic_size	top_n_words	diversity
10	-	-	-	-	-	0.663633	-	-	-	-	10.0	20.0	2.0	1.0
1	10.0	1	1	50.0	0.001	0.609844	-7.953362	0.150283	4.697633	LdaModel<num_terms=2388, num_topics=10, decay=...	-	-	-	-
0	10.0	1	1	100.0	0.001	0.609844	-7.953362	0.150283	4.697633	LdaModel<num_terms=2388, num_topics=10, decay=...	-	-	-	-
3	10.0	1	1	50.0	0.01	0.589140	-7.953418	0.142828	5.137922	LdaModel<num_terms=2388, num_topics=10, decay=...	-	-	-	-
2	10.0	1	1	100.0	0.01	0.589140	-7.953418	0.142828	5.137922	LdaModel<num_terms=2388, num_topics=10, decay=...	-	-	-	-
11	-	-	-	-	-	0.567923	-	-	-	-	10.0	20.0	10.0	0.8
12	-	-	-	-	-	0.545355	-	-	-	-	10.0	10.0	2.0	1.0
13	-	-	-	-	-	0.545355	-	-	-	-	20.0	20.0	2.0	1.0
14	-	-	-	-	-	0.545355	-	-	-	-	20.0	10.0	2.0	1.0
15	-	-	-	-	-	0.545355	-	-	-	-	15.0	10.0	2.0	1.0
16	-	-	-	-	-	0.545355	-	-	-	-	15.0	20.0	2.0	1.0
5	10.0	1	0.01	50.0	0.001	0.503292	-52.725953	0.17547	1.380924	LdaModel<num_terms=2388, num_topics=10, decay=...	-	-	-	-
4	10.0	1	0.01	100.0	0.001	0.503292	-52.725953	0.17547	1.380924	LdaModel<num_terms=2388, num_topics=10, decay=...	-	-	-	-
6	10.0	1	0.01	100.0	0.01	0.496313	-52.763166	0.176814	1.37243	LdaModel<num_terms=2388, num_topics=10, decay=...	-	-	-	-
7	10.0	1	0.01	50.0	0.01	0.496313	-52.763166	0.176814	1.37243	LdaModel<num_terms=2388, num_topics=10, decay=...	-	-	-	-
8	5.0	asymmetric	1	100.0	0.01	0.495511	-7.440948	0.756774	0.374434	LdaModel<num_terms=2388, num_topics=5, decay=0...	-	-	-	-
9	10.0	0.5	auto	100.0	0.01	0.494594	-7.725926	0.381465	0.864773	LdaModel<num_terms=2388, num_topics=10, decay=...	-	-	-	-
17	-	-	-	-	-	0.449230	-	-	-	-	10.0	10.0	5.0	1.0
18	-	-	-	-	-	0.434543	-	-	-	-	15.0	20.0	10.0	1.0
19	-	-	-	-	-	0.415843	-	-	-	-	15.0	10.0	10.0	0.866667

Figure 17: Details results of all 3 tunned models the highest coherence score is achieved by tunned BERTopic. The above results concern the application of the models on the Customer Support Dataset.

	num_topics	alpha	eta	iterations	gamma_threshold	coherence	perplexity	silhouette_score	davies_bouldin_index	lda_model	nr_topics	min_topic_size	top_n_words	diversity
10	-	-	-	-	-	0.821978	-	-	-	-	10.0	10.0	2.0	0.85
11	-	-	-	-	-	0.820739	-	-	-	-	20.0	10.0	2.0	0.8125
12	-	-	-	-	-	0.803961	-	-	-	-	15.0	10.0	2.0	0.766667
13	-	-	-	-	-	0.755583	-	-	-	-	10.0	20.0	2.0	0.722222
14	-	-	-	-	-	0.718228	-	-	-	-	15.0	20.0	2.0	0.642857
1	10.0	1	1	50.0	0.001	0.644640	-8.316868	0.336871	0.584514	LdaModel<num_terms=4309, num_topics=10, decay=...	-	-	-	-
0	10.0	1	1	100.0	0.001	0.644640	-8.316868	0.336871	0.584514	LdaModel<num_terms=4309, num_topics=10, decay=...	-	-	-	-
3	10.0	1	1	50.0	0.01	0.641850	-8.316962	0.396844	0.619592	LdaModel<num_terms=4309, num_topics=10, decay=...	-	-	-	-
2	10.0	1	1	100.0	0.01	0.641850	-8.316962	0.396844	0.619592	LdaModel<num_terms=4309, num_topics=10, decay=...	-	-	-	-
15	-	-	-	-	-	0.622518	-	-	-	-	20.0	20.0	2.0	0.5
16	-	-	-	-	-	0.526119	-	-	-	-	15.0	10.0	5.0	0.8
17	-	-	-	-	-	0.518171	-	-	-	-	10.0	10.0	5.0	0.82
4	5.0	asymmetric	1	50.0	0.01	0.502020	-7.83957	0.763992	0.317717	LdaModel<num_terms=4309, num_topics=5, decay=0...	-	-	-	-
5	5.0	0.1	1	100.0	0.001	0.491193	-7.849722	0.78189	0.310819	LdaModel<num_terms=4309, num_topics=5, decay=0...	-	-	-	-
18	-	-	-	-	-	0.487872	-	-	-	-	20.0	20.0	5.0	0.7
6	5.0	asymmetric	1	50.0	0.001	0.482654	-7.839712	0.765512	0.325026	LdaModel<num_terms=4309, num_topics=5, decay=0...	-	-	-	-
19	-	-	-	-	-	0.478154	-	-	-	-	20.0	10.0	5.0	0.723077
7	5.0	0.1	1	100.0	0.01	0.470084	-7.847911	0.803867	0.2986	LdaModel<num_terms=4309, num_topics=5, decay=0...	-	-	-	-
8	5.0	asymmetric	1	100.0	0.001	0.468894	-7.839851	0.790618	0.295123	LdaModel<num_terms=4309, num_topics=5, decay=0...	-	-	-	-
9	5.0	0.01	1	100.0	0.01	0.467947	-7.816204	0.841657	0.31311	LdaModel<num_terms=4309, num_topics=5, decay=0...	-	-	-	-

Figure 18: Details results of all 3 tunned models the highest coherence score is achieved by tunned BERTopic. The above results concern the application of the models on the Covid Data set.

Additional, details regarding the evaluation and parameter tunning for the BERTopic specifically, since it was the highest scoring model can be found in Appendix VII

4.4. Limitations

The primary limitations of this approach relate to the computational resources required, which essentially translate to the time needed to train the models on the dataset. For instance, training an LDA model without any tuning on 10,000 data points took approximately 26 mins while trials with the same size while applying sets of hyperparameters for the same dataset proved to take several hours for one model. Consequently, we were restricted to using a sample size of no more than 1,000 data points for model tuning, especially given that we needed to train and tune three different models. This constraint prevented us from thoroughly testing the models' performance in a more realistic scenario that would involve a significantly larger dataset.

Besides the translation we also tried to implement a summarization on a sample dataset by using an encoder-decoder architecture, incorporating Long Short-Term Memory (LSTM) layers and an attention mechanism. More information can be found in Appendix VII.

5. Named Entity Recognition

5.1. Overview

Named Entity Recognition (NER) effectively constitutes a problem of entity classification to various labels/classes defined by researchers in a domain of interest (Lample, Ballesteros, Subramanian, Kawakami, & Dyer, 2016). The research community has been dealing with this task for several decades and has developed various approaches over the years (see Appendix III), aiming to surpass the challenges imposed by its requirements in domain-knowledge and the lack of sufficient volumes of labeled data to train effective and accurate models on (Luiggi, Soulier, Guigue, Jendoubi, & Baelde, 2023).

5.2. Methodology

We implemented various approaches representative of the SOTA techniques along most phases of NER's evolution. We have discarded rules-based approaches as they showcase little generalization capacity compared to modern systems.

5.2.1. Text Preprocessing

We have implemented minimal preprocessing by casting all texts to the lower case and removing URLs, which bear no contextual meaning.

5.2.2. Annotation Schema

We decided to recognize entities of the following types (or 'labels') among the texts of the tweets from the 'uns_covid1' dataset:

- 'generic'
- 'medical-conditions'
- 'symptoms'
- 'treatments'
- 'organizations'
- 'policies'
- 'metrics'

For detailed breakdown of labels considered, together with an explanation regarding the tag schema used and how annotation was implemented, see Appendix IV.

5.2.3. Research mentality - assumption

We took label/class imbalances into account by optimizing for the 'class_weights' parameter of the 'LogisticRegression' class of scikit-learn. In the other approaches, no similar step was taken.

That was deliberate; since we had a vast amount of annotated samples, we could have selectively extracted a training set being as close to being perfectly balanced as possible.

Figure 19: Distribution of Label Frequency in Annotated Training Data by Class (in 100,000 Training Samples)

However, this cannot be an option under settings where limited human resources are available and manual annotation needs to take place due to the risk involved in cases of mislabeling. Having so many annotated samples is also impossible when working with low-resource languages where there cannot even exist a sufficient number of training samples so as to select a balanced set out of them.

Trying to simulate an environment of limited availability in annotated data and comparing the models' performance in it, we decided to take into account only 500 from our annotated samples for training. This does not apply to the PromptNER model; There we detailed only 3 few-shot samples.

5.3. Architectures and Methods

5.3.1. Statistical NER

We have selected to implement the Maximum Entropy (ME) Model as presented in (Borthwick, 1999) based on the implementation of the 'LogisticRegression' class in scikit-learn (Pedregosa et al., 2011).

This model performs token label classification based on information on the surrounding tokens as well as other (linguistic, like POS-related) features a user might have extracted. We only considered token-level labels, however, since we wanted to make comparisons between the models on equal terms, with the rest of the models (as do DL- and Transformer- based models in general) only considering token-level information

It calculates the weighting to be applied to each feature as part of estimating as part of the identification of a probability function that maximizes entropy (so it is essentially as uniform as possible) while still fitting to the observed data.

This consistency with the training data results in the ME distribution aligning with the Maximum Likelihood distribution, making it less prone to overfitting, since it does not make any assumptions that are not founded on the training set. As such, it promises good generalization performance.

5.3.2. Architectures and Methods – Transformer-Based NER

We trained several NER pipelines based on the capabilities offered by the SpaCy library to harness the power of their own developed transformer models. SpaCy allows for eight different approaches in NER pipeline construction, offering options on the following:

- Algorithms optimized for CPUs or GPUs.
- Algorithms optimized for efficiency or accuracy.
- Training a model from scratch or retraining an existing one offered by SpaCy.

We tried all these approaches, expecting the retraining options to be both more efficient and accurate under a setting of limited computational resources. We opted for the retraining of the 'en_core_web_sm' model which is smaller in size and promised faster retraining execution time.

5.3.3. Architectures and Methods - LLM-based NER

Our team decided to implement an LLM-based NER pipeline to testify to its ability to give accurate results under limitations in available resources, time to train models, as well as annotated data. To this point, we developed an implementation of PromptNER (Ashok & Lipton, 2023) and as the

underlying LLM we opted for DistilBERT (Sanh, Debut, Chaumond, & Wolf, 2020), which is an implementation of BERT focused on preserving computational resources and boosting training.

5.4. Results

Since we treat NER as a case of text classification (even on the token-level), we assess the performance of our models based on relevant metrics. We noticed that our training data contained severe class imbalances. This is why we ignored the metric of accuracy and opted for precision, recall, and the F1-Score to evaluate our models.

It is important to note that for all these metrics we considered their macro averages across all classes when considering overall model performance so as to prevent falsely influencing their values by the sample size of the better represented (and more successfully predicted) labels. Furthermore, all models were assessed against the very same testing set of 10,000 samples.

5.4.1. Statistical NER

```
Best parameters found:  {'C': 10, 'class_weight': None, 'penalty': 'l2'}
Best cross-validation macro F1 score:  0.23217996903455007
Test Accuracy: 0.94
                    precision    recall  f1-score   support

           B-GENERIC       0.86      0.06      0.11     14461
B-MEDICAL-CONDITIONS       0.00      0.00      0.00        23
           B-METRICS       0.77      0.44      0.56      2042
      B-ORGANIZATION       0.82      0.42      0.55       204
          B-POLICIES       0.03      0.01      0.01       717
          B-SYMPTOMS       0.00      0.00      0.00        50
        B-TREATMENTS       0.02      0.01      0.01       189
           I-GENERIC       0.91      0.80      0.85     15691
I-MEDICAL-CONDITIONS       0.00      0.00      0.00         3
           I-METRICS       0.00      0.00      0.00       499
      I-ORGANIZATION       0.00      0.00      0.00        23
          I-POLICIES       0.00      0.00      0.00       222
          I-SYMPTOMS       0.00      0.00      0.00        22
        I-TREATMENTS       0.00      0.00      0.00         4
                   0       0.94      1.00      0.97    292145

            accuracy                           0.94    326295
           macro avg       0.29      0.18      0.20    326295
        weighted avg       0.93      0.94      0.92    326295
```

Figure 20: Performance metrics per class and overall for the optimized Maximum Entropy Model

As we expected, the model did pretty well with the highly-represented classes, achieving a recall of 1 for the '0' class, higher than its precision of 0.94 for the same class (showing bias towards classifying a token as such more easily than it did for other classes). It was also rather successful on 'I-GENERIC' and 'B-ORGANIZATION' tokens (organizations were typically expressed with a single word), which made for the best-represented labels in the training data, in contrast to under-represented classes on which it completely failed at capturing any of the actual labels on the testing set.

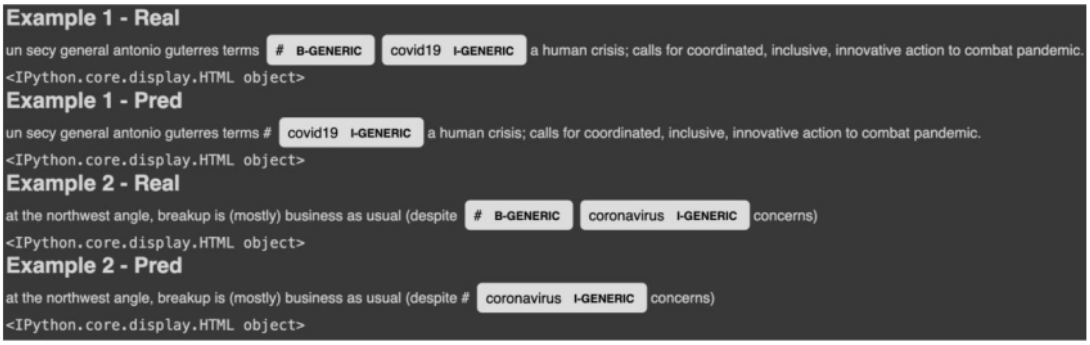

Figure 21: Sample Annotations Produced by the Maximum Entropy Model

5.4.2. Transformer-Based NER

Table 5: Macro Average Model-Level Performance Metrics for the SpaCy Models (Re-)Trained

config_file_name	training_n	overall_precision	overall_recall	overall_f1
base_config_en_gpu_accuracy_null.cfg	500	0.96	1	0.98
base_config_en_cpu_efficiency_retrain.cfg	500	0.95	1	0.97
base_config_en_cpu_efficiency_null.cfg	500	0.95	1	0.97
base_config_en_cpu_accuracy_null.cfg	500	0.92	1	0.96
base_config_en_cpu_accuracy_retrain.cfg	500	0.92	1	0.96
base_config_en_gpu_efficiency_null.cfg	500	0.92	1	0.95
base_config_en_gpu_efficiency_retrain.cfg	500	0.91	1	0.95
base_config_en_gpu_accuracy_retrain.cfg	500	0.8576132895	1	0.9114099413

We observe a significant improvement in generalization performance from the macro average F1 score 0.20 of the optimized Maximum Entropy model to a minimum of 0.91 with less than 2 hours of (re-)training of each SpaCy model we configured.

We also see that the gpu-optimized model focused on prediction accuracy and taking no default spacy model into account, performing the best on a TPU rented from Google, did the best work on capturing the contextual meaning of our samples. And it succeeded in a setting of limited available samples and pre-processing, staying true to the promises of the attention mechanism (Vaswani et al., 2017). Its good performance even on the very misrepresented classes testifies to the previous.

Table 6: Class-Level Performance Metrics for the Optimal SpaCy Model

Label	Precision	Recall	F1
GENERIC	0.99	1	0.99
METRICS	0.96	1	0.98
TREATMENTS	1	1	1
MEDICAL-CONDITIONS	1	1	1
ORGANIZATION	0.95	1	0.98
POLICIES	0.85	1	0.92
SYMPTOMS	1	1	1

Figure 22: Sample Annotation Produced by SpaCy's Optimal Model

5.4.3. PromptNER

Our implementation worked as it indeed produced labels. However, all of them identified with the best-represented class, which is '0', resembling non-named entities.
This hints at potential issues in the prompt used or the limitations of the underlying LLM when used without retraining.

Table 7: Performance of PromptNER Model per-Class and Across

Metric	precision	recall	f1	support
CONDITIONS	0	0	0	2
GENERIC	0	0	0	113
METRICS	0	0	0	12
POLICIES	0	0	0	10
TREATMENTS	0	0	0	1
SYMPTOMS	0	0	0	5
across_prec	0	0	0	0
across_rec	0	0	0	0
across_f1	0	0	0	0
across_acc	0.9	0.9	0.9	0.9

5.5. Conclusions and Future Work

Transformers and transfer learning via pre-trained models has significantly expanded not just the prediction capabilities but also the research horizons in NER, at the point where we are able to observe such effects in our small-scale experiments. Further experiments could be performed in the future to improve on the showcased architectures (see Appendix V).

6. Text Classification

The goal was to produce a Vectorizer – Classifier system with perfect predictive performance, in an efficient manner, meaning producing this within a sensible time frame. It wouldn't have been possible to test every combination of models and hypermeters. To that end, we came up with a 3-phase-plan, which allowed for fewer iterations and less processing time overall.

Phase 1. Finding the best combination of Vectorizer – Classifier
By using default parameters or small parameter ranges to grid-search for optimization, as well as small training sizes, we could ascertain which models are the most promising for the next steps.

Phase 2. Finding the optimal hyperparameters
At this point, we would already know which Vectorizer – Classifier pair to use for a dataset, and we would only need to find the optimal hyperparameters for them. Large training sizes and parameter ranges can be used, since the training and validation would be done for a single pair of Vectorizer and Classifier.

Phase 3. Feature selection
Features created from the preprocessing, as well as generated from the NER algorithms, can be part of the classification training and validation sets. But due to those features being numerous, a selection of the optimal set is in order. At this point, we could have the optimally parameterized Vectorizer – Classifier pair, and so performing training and validation for different sets of features, combined with feature importance information produced from the model, would lead us to the optimal set of complimentary feature set for our classification.
Due to time constraints, the above was modified to the below process.
Phase 1.
We used medium-sized parameter ranges to grid-search for optimization, as well as small training sizes.
Phase 2.
We used the most accurate models, with their optimized parameters and the text-related features when advisable, to make predictions for each dataset.

Even though the former plan would have been more efficient and have produced the most accurate models, the latter provided fairly accurate models in a shorter time frame.

6.1 Models and Parameters

For the models used for vectorization and classification, and their parameter ranges, the selections were based on the available literature and best practices for text classification (Brownlee, 2020):

Vectorizers

- Count
- Tf-Idf
- FastText

The FastText model is a Word2Vec variant that allows for vectorization and classification. It was selected over the Word2Vec model due to it creating vectors of uniform size, in contrast to Word2Vec, a peculiarity that fitted the optimizing algorithms that were already set up. The FastText model was used exclusively for vectorization.

Classifiers

- Logistic Regression
- Support Vector Machine
- Bagging Classifier
- Random Forest
- Gradient Boosting

A collection of linear, probabilistic and statistical, and ensemble models.
Pretrained BERT models were also utilized, but their use was limited to benchmarking tests. More details on that in the relevant chapter.

Table 8: Parameter Ranges for Vectorization and Classification Models

Model	Max df	Max features	Max df	Min df	Ngram range	norm	C	Solver	Kernel	N-estimators	Max depth	Learning rate
TF-IDF	0.25, 0.5	None, 5000, 10000	-	-	(1, 2), (1, 3)	l1, l2, None	-	-	-	-	-	-
COUNT	-	None, 100, 1000	0.5, 0.75, 1	0.1, 1, 5	(1, 2), (2, 2)	-	-	-	-	-	-	-
LogisticRegression	-	-	-	-	-	-	100, 10, 1, 0.1	newton-cg, lbfgs, liblinear	-	-	-	-
SVM	-	-	-	-	-	-	50, 10, 1	-	poly, rbf, sigmoid	-	-	-
Bagging CLF	-	1, 0.3, 0.1	-	-	-	-	-	-	-	10, 100	-	-
Random Forest	-	sqrt, log2	-	-	-	-	-	-	-	10, 100	-	-
Gradient Boosting	-	-	-	-	-	-	-	-	-	10, 100	3, 5	0.01, 0.1

Given the above choices of models, the parameter ranges used to grid-search for hyperparameters are as follows:

Originally the parameter ranges were much larger but were limited to the above due to time limitations.

6.2. Results

6.2.1. CPU – TPU system benchmarking

Since the available resources were a default CPU system and a multi-core, very high memory system (96 cores and 334 GBs of RAM), tests were performed to ascertain the efficiency of these options.

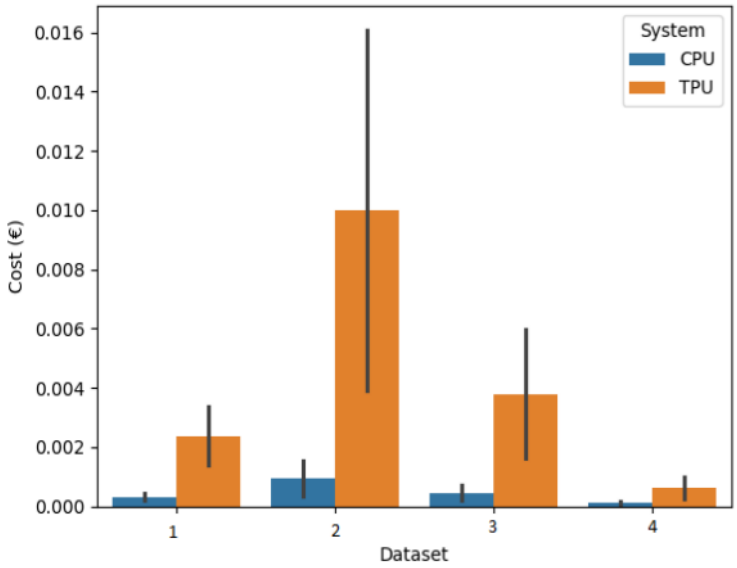

Figure 23: Average tuning cost of CPU and TPU systems for four datasets

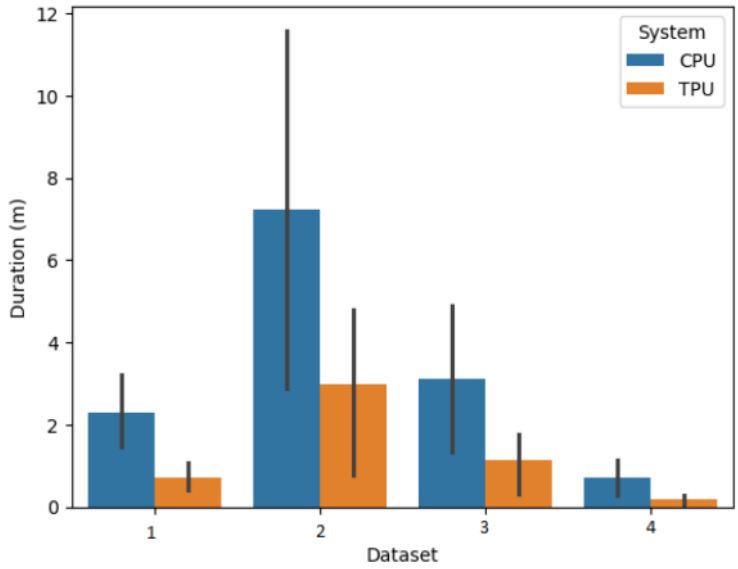

Figure 24: Average tuning processing time for CPU and TPU systems for four different datasets

The above figures demonstrate a comparison of average speed and cost of these options for optimizing four different datasets. The TPU system appears to be on average two times faster than the CPU system, but, on average, over 5 times more costly than a regular CPU system. Due to the short processing time required for our purposes, the cost was not a factor, therefore the TPU system was selected for optimization and selection of the models used for predictions.

6.2.2 Tuning Results

The below figures show the accuracy results of classification for three different datasets with only text vectors and features. The results are sectioned depending on the text preprocessing, size of training size, and on the second figure, the vectorizing model used.

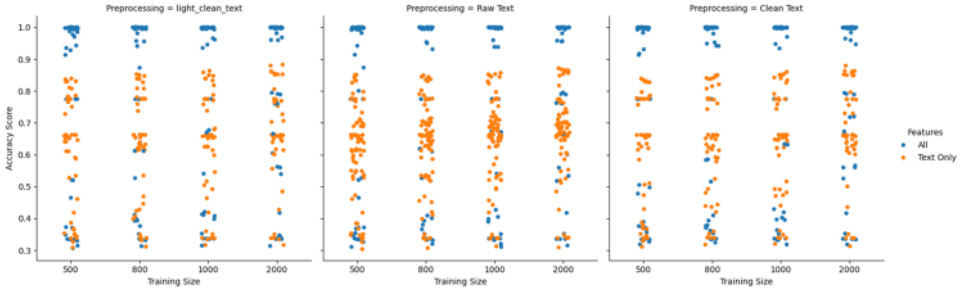

Figure 25: Tuning results with various parameters

Figure 26: Tuning results for various parameters, datasets and models

The use of features, generated from the NER processes, proved to increase the predictive performance by a significant margin, for every model and for most datasets. As mentioned before, a feature selection process has the potential to improve the performance of classification models even further.

The different training sizes were used to ascertain the potential for predictive performance of each model, meaning that models that scaled better with increased training sizes would have been selected. The result showed a linear relationship, with a very small slope, between training size and accuracy scores.

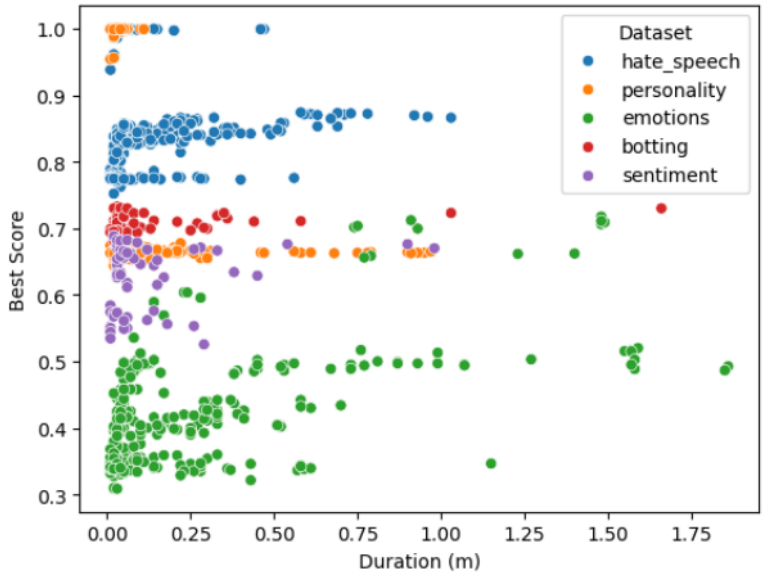

Figure 27: Tuning results; Comparing performance with processing time for each dataset

The above figure demonstrates that for all datasets, models that required longer processing time to tune, do not outperform those that require less processing time. This suggests that careful preparation of parameters and model choices can significantly reduce the tuning phase of creating a classification system.

Table 9: Top 3 tuning resullts for each dataset

hate_speech				
Vectorizer	Features	Preprocessing	Accuracy	Duration (s)
CountVectorizer	All	Clean	1.0	0.6
CountVectorizer	All	light_clean_text	1.0	0.8
TfidfVectorizer	All	Clean	1.0	1.3

personality				
Vectorizer	Features	Preprocessing	Accuracy	Duration (s)
CountVectorizer	All	light_clean_text	1.0	14.3
CountVectorizer	All	Clean	1.0	14.4
CountVectorizer	All	Clean	1.0	14.5

emotions				
Vectorizer	Features	Preprocessing	Accuracy	Duration (s)
TfidfVectorizer	Text	light_clean_text	0.7687	18.8
TfidfVectorizer	Text	Raw Text	0.7642	18.8
CountVectorizer	Text	light_clean_text	0.7637	25.6

botting				
Vectorizer	Features	Preprocessing	Accuracy	Duration (s)
TfidfVectorizer	Text	Default	0.7338	31.1
CountVectorizer	Text	Default	0.7335	22.2
CountVectorizer	Text	Default	0.7329	25.8

sentiment				
Vectorizer	Features	Preprocessing	Accuracy	Duration (s)
TfidfVectorizer	Text	Default	0.7015	461.9
CountVectorizer	Text	Default	0.7013	28.1
TfidfVectorizer	Text	Default	0.6871	30.2

The table above shows the top performing models and parameters, features and preprocessing algorithms, for each dataset. The most noteworthy result is for the sentiment dataset, which shows that the most accurate combination required two hundred times longer tuning compared to the second most accurate, but the accuracy gain was less that 1%.

6.2.3. BERT pretrained models

Regarding BERT pretrained models, the tests were isolated to a single dataset. BERT models were used to tokenize and classify tweets and the results were recorded. Another test was performed by tokenizing tweets, later the tokenized tweets and relevant labels were used to fine-tune the pretrained models, and again the associated classification results were recorded.

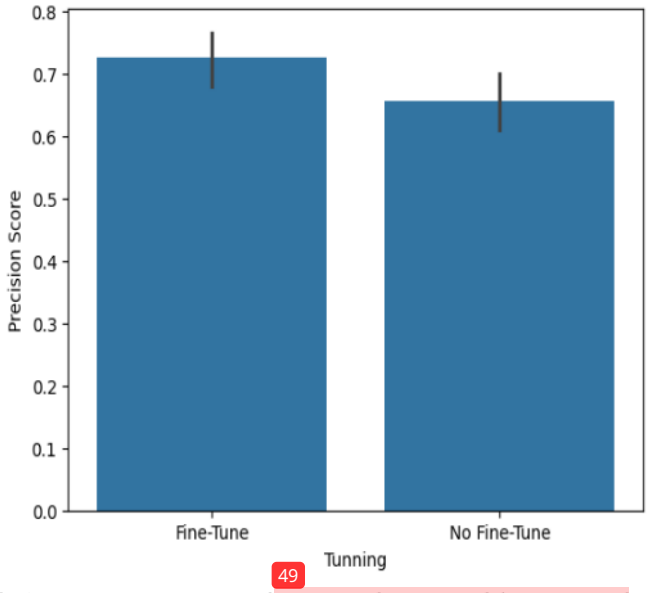

Figure 28: Average precision score of pre-trained BERT models; Fine-tuned and default

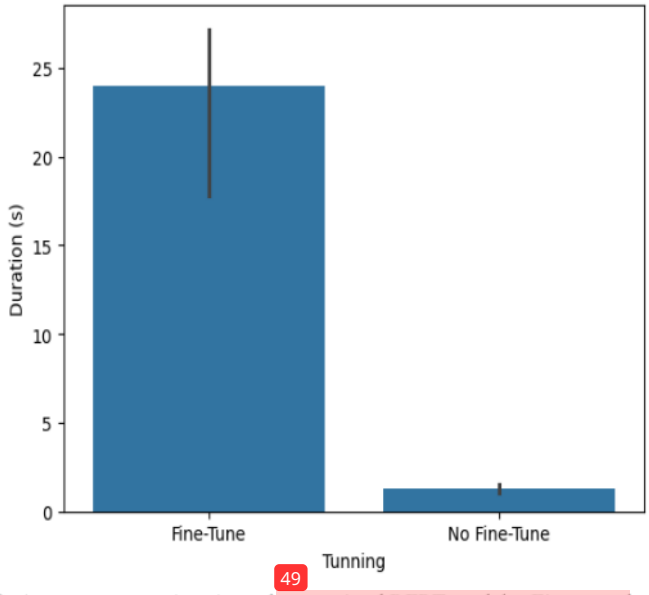

Figure 29: Average processing time of pre-trained BERT models; Fine-tuned and default

The results from the above figures show that the small performance increase from pretraining does not justify the major processing time requirements of fine-tuning the pretrained models. In general, the use embedding models for classification appear to be too excessive for the complexity of such

problem, considering the processing time required by more simplistic models, which perform the same or for some cases outperform the more sophisticated embeddings models.

6.3. Final Models and Applications

Table 10: Descriptions and scores for the final models used for each dataset

Dataset	Preprocessing	Features	Vectorizer	Classifier	Duration (s)	Peak Memory (GB)	Accuracy Score	F1 Score	Precision Score	Recall Score
hate_speech	clean_text	All	CountVectorizer	LogisticRegression	1.8	8.24	1	1	1	1
Emotions	light_clean_text	Text Only	TfidfVectorizer	GradientBoosting	61.0	8.41	0.8474	0.8478	0.8651	0.8474
personality	light_clean_text	All	CountVectorizer	SVC	27.6	8.95	0.9999	0.9999	0.9999	0.9999
botting	text	Text Only	TfidfVectorizer	LogisticRegression	24.7	10.15	0.7845	0.7757	0.7760	0.7845
sentiment	text	Text Only	TfidfVectorizer	LogisticRegression	99.8	11.63	0.7955	0.7955	0.7957	0.7955

The above table shows the various scores for the models that were used for each dataset's final predictions. The tuned vectorizer and fitted classifier of the botting dataset were used to transform and classify the datasets of Covid-19 and Ukraine War tweets.

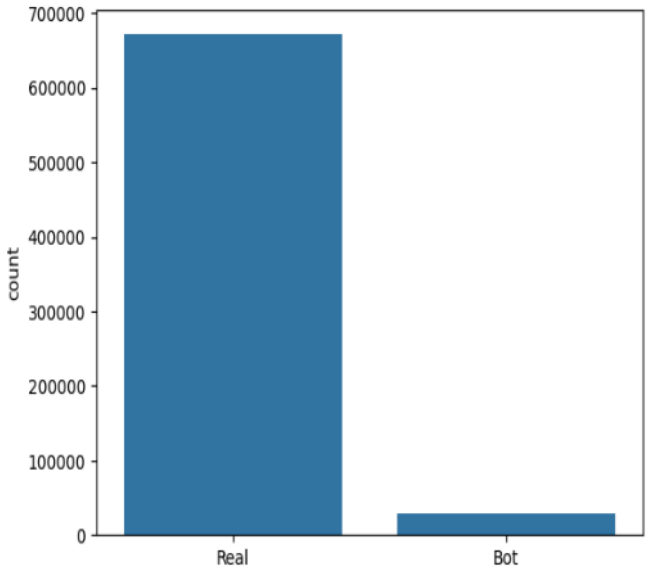

Figure 30: Bar plot of botting classifier applied to the COVID-19 dataset

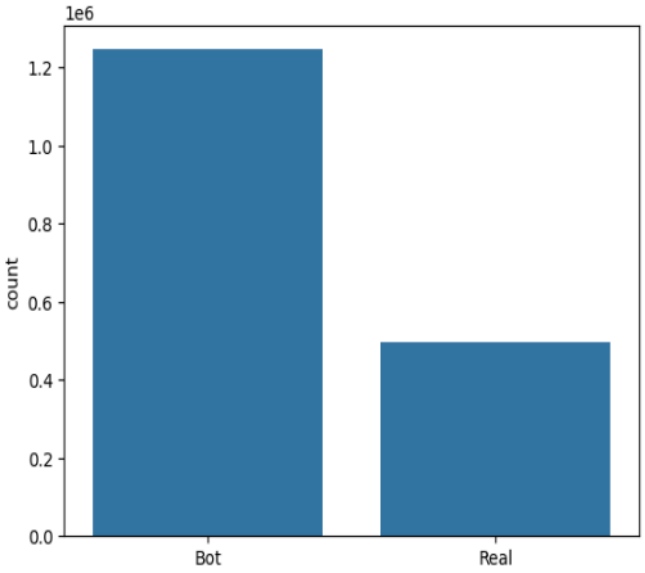

Figure 31: Bar plot of botting classifier applied to the Ukraine War dataset

Considering that the accuracy score of the fitted classifier was close to 80%, and even assuming a false-positive rate of 50%, classifying over half of the Ukraine War tweets as tweets made from bots, is an indication for a sad reality and the general state of technology utilization on social-media platforms.

6.4. Text Classification XAI

The SHAP XAI technique was applied to the botting and the personality classifiers, on a sample of each dataset.

Text: “unmasking probe commission barr concludes charge public report october non surprise” – Label: Real User

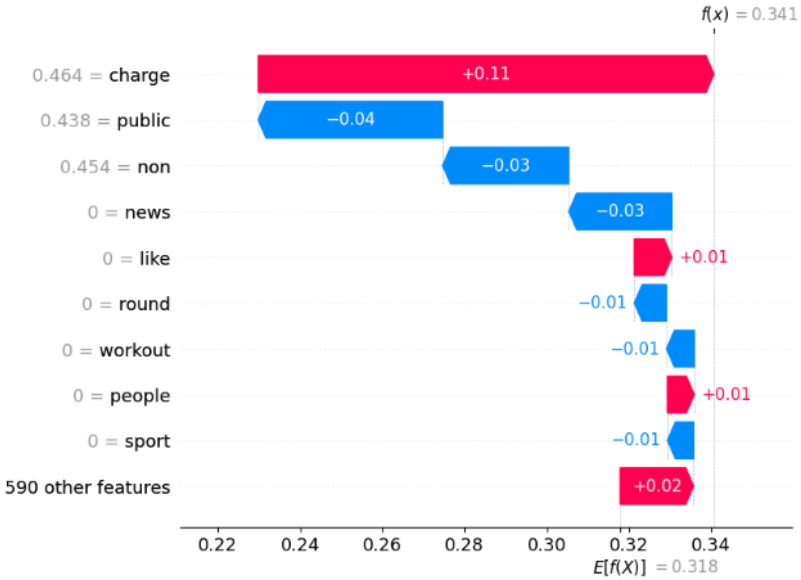

Figure 32: SHAP values for text from actual user

Text: “playing genius motherfu feat chip gt” - Label: Bot

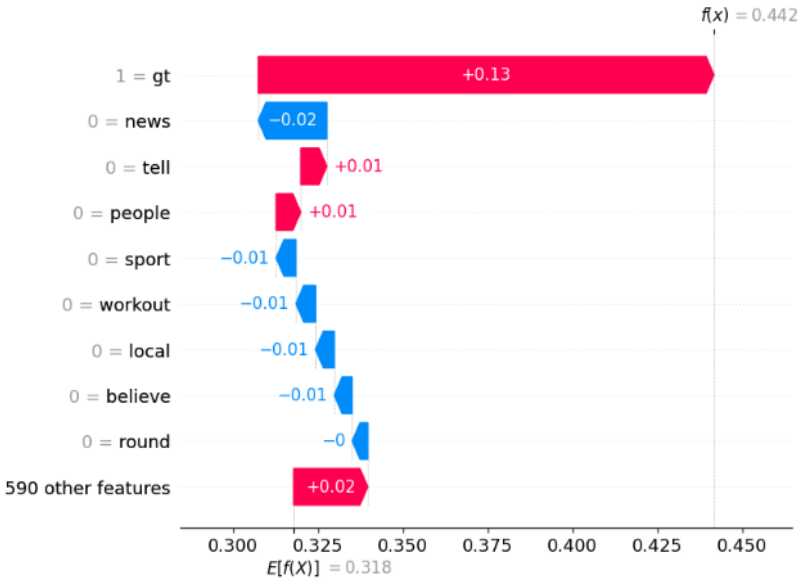

Figure 33: SHAP values for text from a bot

The above figures show the significance of terms like “charge” and “gt” for the classification of a tweet’s origin being a bot, and terms like "news” and “public” being an actual twitter user.

By testing the following:
Text: “who thought they were straight from the start lets be real “ - Label: Introvert

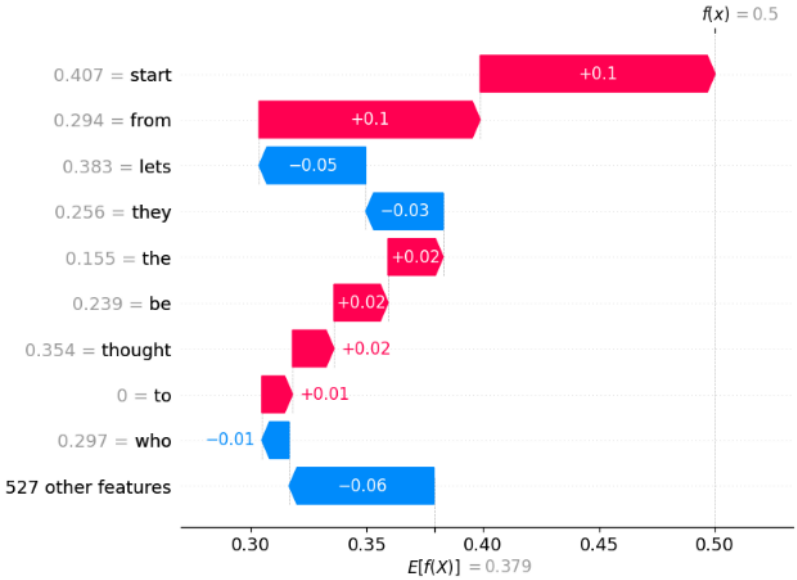

Figure 34: SHAP values for text from introvert

Text: “i am going to kiss u both” - Label: Extrovert

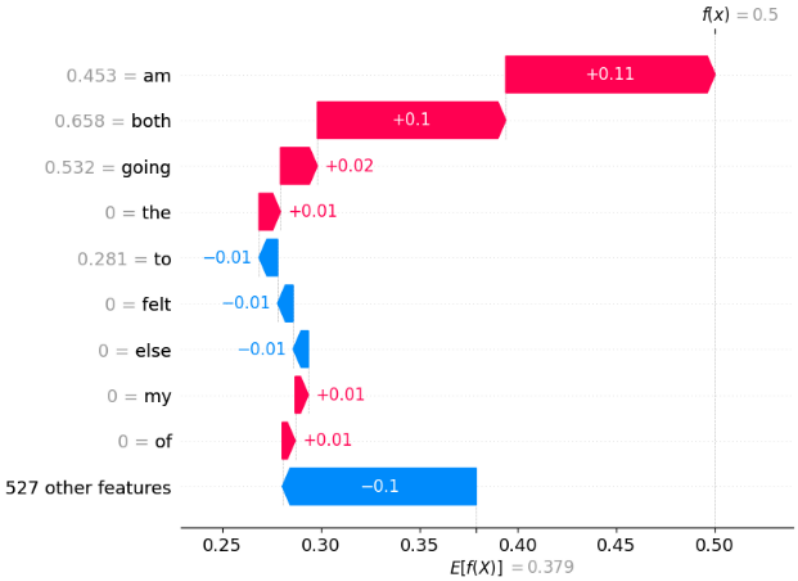

Figure 35: SHAP values for text from extrovert

Terms like “start”, “am” and “both” have a major contribution to classifying a user as an extrovert, while terms like "lets” and “they” lead the classifier to label the user as an introvert.

7. Machine Translation

7.1. Overview

Machine Translation (MT) is the field of Machine Learning dealing with the automated translation of text from one natural language to another. Similar to NER, it has followed an evolutionary line beginning from rules-based approaches which mainly consisted of hard-coded solutions. In (Hutchins, 2004) we find a reference to the work of an inventor whose work became known in 1933. Georges Artsrouni-constructed a device that acted as a mechanical dictionary for word-to-word translations, whose components are still evident (yet evolved) in any modern MT system:

- Input/output sequences (single words in this case) are cached in a 'memory' that could hold up to 40,000 words.
- Words were keyed-in (or 'indexed) upon input.
- Efforts were made to build an efficient search mechanism to match between words in the two languages.

Multiple researchers followed. Even starting from the 1950s, researchers like Trojanskij were pondering on the challenge of natural language ambiguity. Phrase-based statistical techniques are found in SOTA applications of the 2000s, later becoming more efficient in their training and predictions through the introduction of DL-based approaches. The latter completely shifted the research landscape after the introduction of the attention mechanism (Vaswani et al., 2017), which paved the way to the development of LLMs and their adoption in MT applications. For a more detailed timeline, linked to our project's scope and goals, see Appendix VI.

7.2. Methodology

We tackled this challenge with both bare models that needed training from scratch as well as harnessing the power of pre-trained models freely available online. The reasoning behind this was to showcase the implementation of the SOTA architectures in machine translation in the most recent evolutionary stages of the field.

7.2.1. Architectures and Methods - Sequence-to-Sequence Learning with Neural Networks

7.2.1.1. Text Preprocessing

The text preprocessing implemented was minimal, consisting of text standardization by casting texts in the lower case. Then, we vectorized the data using T5's vectorizer. It is important to mention that we manually handled the vocabulary size, so we removed the corresponding limitation in our code and dictated that we will work with a 5,000-token-long vocabulary.

7.2.1.2. Encoder-Decoder Stacks

We opted for the use of Gated Recurrent Units (GRU) both in the encoder as well as the decoder stacks of the two models we considered. This was a choice driven by the need for computational efficiency in view of our limited available resources, but it also served as a good opportunity to improvise on online tutorials that revolved around LSTM-based architectures (Chollet, 2017).
The number of units was 256 in total, which is a typical choice in the community and in bibliography and adds enough depth to the model so as to harness the improvement first showcased by (Sutskever et al., 2014). We chose not to include an attention layer in this implementation due to the fact that we also used a transformer-based method which includes a SOTA attention mechanism.

7.2.1.3. Regularization

To enhance efficiency and prevent overfitting in training these models, we took two kinds of measures: first, we implemented Lasso (or 'L1') regularization so as to suppress the trained weights from inflating too much; we then employed the method of early stopping, so as to prevent the model from continuing to train if further training does not assist in improving its generalization performance (no validation loss reduction for at least five consecutive epochs). We also padded all input/output sequences to the maximum length the tokenizer came across for each of them, so as to expect sequences of a specific length. The vocabulary size considered was also rather short at around 5,000 words.

7.2.1.4. Optimizer

We utilized the Adam algorithm as it is typically recommended as the default algorithm to try in an application.

7.2.1.5. Loss Function

Sparse Categorical Cross Entropy was considered. Categorical cross-entropy is a probabilistic loss performance metric used in cases where there are two or more label classes (Keras, n.d.). We chose the 'sparse' implementation of it to increase its efficiency when the labels are given as integers (such as in our case where real and predicted output tokens are indexed).

All the previous brought the parameters to tune to 25,474,404 in total.

Table 11: Summary of our Seq2Seq Model

Layer (type)	Output Shape	Param #	Connected to
encoder_inputs (InputLayer)	[(None, None)]	0	[]
decoder_inputs (InputLayer)	[(None, None)]	0	[]
encoder_embedding (Embedding)	(None, None, 256)	8217600	['encoder_inputs[0][0]']
decoder_embedding (Embedding)	(None, None, 256)	8217600	['decoder_inputs[0][0]']
encoder_gru (GRU)	[(None, 256),	394752	['encoder_embedding[0][0]']
	(None, 256)]		
decoder_gru (GRU)	[(None, None, 256),	394752	['decoder_embedding[0][0]',
	(None, 256)]		encoder_gru[0][1]']
decoder_dense (Dense)	(None, None, 32100)	8249700	['decoder_gru[0][0]']
Total params	**25474404 (97.18 MB)**		
Trainable params	**25474404 (97.18 MB)**		
Non-trainable params	**0 (0.00 Byte)**		

7.2.2. Architecture and Methods - Pretrained Transformer-based model

The wide-spread respect that freely available pre-trained models have today rendered the training of our own transformer model from scratch a far-fetched venture. Thus, we opted for the prospect of using Google's 'T5: the Text-To-Text Transfer Transformer', which is a SOTA pretrained encoder-decoder transformer model promising excellent results after retraining on case-specific environments even with limited resources available, as its architecture is concentrated on transfer learning cases (Raffel et al., 2019).

7.2.3. Evaluation

The Bilingual Evaluation Understudy (BLEU) metric was used to assess the model-generated translations, based on validation samples drawn from the WMT 2014 English-German dataset (Lotfy, n.d.).

We took into consideration a maximum clipped n-gram precision rank of 4 to bring our implementation as close to the original paper (Papineni, Roukos, Ward, & Zhu, 2002) as possible. We assigned equal weights to calculations for all n-grams, as the BLEU scores generally tend to drop as the sentence length increases.

We took into account the same metric for all models to ensure that we make comparisons on the same basis.

7.3. Results

As expected, the pretrained model had the best performance even after retraining on a very limited corpus of 50,000 samples, which the attention-based model under our detailed architecture did not manage to extract effectively any meaning out of.

On each case, we compared translations of the same sample texts from WMT to calculate the BLEU scores and from the 'uns_covid_1' dataset to make presentable model-by-model comparisons. We showcase the latter sample results together with a reference translation obtained through the DeepL API for machine translation (DeepL, n.d.).

Table 12: BLEU Score Evaluation of MT Models Considered

Model	n_retrain	BLEU Score WMT 2014 references)
Seq2Seq (no attention mechanism)	50,000	0.0003
T5 - no retraining	-	0.09
T5 - after retraining	50,000	0.08

We see a vast increase in performance with the T5 model even though (re)training time was much shorter than with the seq2seq model. It is noteworthy that T5's performance drops against translating WMT 2014 references after retraining, but only 3 epochs were allowed so it is reasonable since it was not given the chance to optimize on the WMT dataset's characteristics.

Table 13: Sample Translation of Text from 'uns_covid1' Dataset

	Input & Reference Translations
Input Text	Guyana: Aviation authority extends suspension of all incoming international flights until 1 May because of #COVID19
Reference Translation (DeepL)	Guyana : Luftfahrt Behörde erweitert Aufhängung von alle eingehend international Flüge bis 1 Mai denn von # COVID19
Seq2Seq (no attention mechanism)	lamp along Wear veche 0,00 the org improvement clip fizice hier anunțat cloud Autumn -22 Hosting call HY hottest pfel utilis
T5 - after retraining	bahnhof hat die flottenangehörigkeit der frankreich

It is clear that retraining a freely available model returned acceptable results that could be used in actual applications whereas the home-trained model, for the same training time, returned incomprehensible phrases.

7.4. Conclusions and Future Work

The experiments demonstrated that the attention mechanism significantly enhances translation quality, with transfer learning using the T5 model showing the best performance. The BLEU score was a reliable metric for evaluating translations, accounting for both precision and brevity. Future work could explore implementing the local attention approach to further improve efficiency and accuracy.

We could also experiment with optimizing our approach by trying various vocabulary sizes and optimization algorithms, as well as integrating further layers into our models, in order to attain further empirical knowledge on what measures can help boost NMT systems' performance, in parallel with the latest research.

APPENDIX

Appendix I

The first text normalization step taken was the conversion of all text into the **lower case** and the dropping of short words (made of less than 4 characters) to prevent counting-in abbreviations or other initials. We also accounted for their existence as a feature to be analyzed.

Moreover, we expanded contractions (and counted their frequency per sample) since their existence is deemed rather likely in informal language usage settings like on social media.

We dropped all numbers as not conveying significant contextual information. We have decided to keep some symbols that may pertain to numerical values, like currency signs, since they may typically be part of tweets expressing sentiment. When it comes to punctuation, we retained question marks ('?'), exclamation marks ('!') and ellipses ('...') because they are the most relevant to expressing emotions and sentiment (Oxbridge Editing, 2024). We also retained double-quotes because they might be used in directly quoting someone's words, which is possible to be followed by a comment and potential criticism or approval. Similarly for the start ('*') character which is typically used in social media posts to obscure foul language.

Spelling error correction was applied as we observed various such utterances that contribute to the randomness in our data.

We also replaced emojis and emoticons with word representations to account for their contextual meaning in the datasets used for machine learning.

Appendix II

We decided to remove usernames and count their frequency per tweet, making the assumption that tweets directly referring to specific people are more likely to express sentiment. We did the same for hashtags. We kept both types of utterances in the dataset used for NER, as they were typically used to refer to the pandemic in general and popular organizations related to it. URLs and HTML elements were completely removed, as were very short words of less than 2-characters' length.

Stopwords (as identified by NLTK's stopword containers for the english language) were dropped, as were repetitive characters (which we replaced with a double utterance, except for the 'ellipsis' mark that bears significant contextual information and we replaced with an '_ELIPSIS_' value).

After all the previous operations, we also made sure to drop any escaped characters and replace any repetitive whitespace characters with a single space character.

Appendix III

Challenges in the field of NER were first attempted to be tackled by knowledge-based methods that relied on the researchers' domain knowledge and available lexical resources ('entity dictionaries') to hard-core entity annotation rules (Pakhale, 2023). Statistical approaches followed, focused on automatically extracting rules for entity annotation based on labeled training data through the implementation of ML algorithms. This started being refined before 2010 by the gradual adoption of Deep-Learning approaches that concentrated on context and entity encoding through Multi-Layer Perceptron (MLP) models (Gallo, Binaghi, Carullo, & Lamberti, 2008).

The latest stages of NER's evolution revolve around Transformer- and LLM-based approaches that promise to spare researchers from the need for extended annotated data and make training processes more efficient (Vaswani et al., 2017).

Appendix IV

Our annotation schema in detail has as follows:

5. 'generic': high-level mentions to the coronavirus pandemic and its implications, e.g., hashtags like "#coronavirus" on generic tweets.
6. 'medical-conditions': mentions to named health conditions related to covid e.g., "long covid", "adult respiratory distress syndrome".
7. 'symptoms': mentions to symptoms of covid e.g., "cough", "fatigue".
8. 'treatments': mentions to treatments applied to covid patients e.g., "prescription", "vaccination".
9. 'organizations': mentions to organizations popular in the pandemic period e.g., "W.H.O.", "NHS","Pfizer".
10. 'policies': mentions to government policies implemented to curb the pandemic e.g., "mask mandate", "mandatory testing", "social distancing".
11. 'metrics': mentions to numbers/metrics related to the implications of covid e.g., "15,000 cases", "3,000 deaths", "1,000 hospitalizations".

Regarding the tag schema(s) used, some models required a simple schema for the training and validation data based on the character boundaries and the label of named entities in text. Others, however, required the implementation of the IOB scheme, which provides for the application of position-dependent labels to tokens being part of a named entity utterance (Cho, Okazaki, Miwa, & Tsujii, 2013). We have developed scripts to accommodate for both cases and translate between the two.

To spare our team from the labor-intensive and time-consuming task of manually annotating data sample by sample, we developed scripts that make use of sophisticated regular expressions (RegEx) based on a set of representative utterances we came up with using InstructGPT through OpenAI's API (OpenAI, 2021) and WordNet's API through its NLTK implementation (Miller, 1995) for the generation of synonyms. This allowed us to rapidly and accurately recognize and annotate named entities across the entirety of our texts.

Appendix V

In later NER-related research ventures, we would consider retraining the transformers-based 'en_core_web_trf' model which comes with a 50,265 words large vocabulary and is based on the RoBERTa architecture.

As for PromptNER, we are very intrigued to inspect its results over such a few annotated examples, so in the future we will try implementing it with a better LLM and a refined prompt with more advanced few-shot examples in the future, like BERT or some other.

Appendix VI

The knowledge-based techniques used by researches around the world were swiftly refined from rules-based systems into statistical approaches, with the SOTA applications in the early 2000s revolving around phrase-based techniques. According to (Brown et al., 1990), a statistical translation systems are based on techniques computing "language model probabilities", and methods that match input sentences with output sentences that maximize conditional probabilities.

In the 2010s, researchers found useful alternatives to such implementations of machine translation that were not made of multiple disjoint parts, making fine-tuning and maintenance very complex (Bahdanau, Cho, & Bengio, 2015). This was tackled by Neural Machine Translation (NMT) techniques, which harnessed the power of neural networks to build models that take a single piece of text, and, through -arbitrarily- complex computations, derive a single output sequence as part of a single encoder-decoder structure (Sutskever et al., 2014).

Moving on to 2017, however, something that no method had managed to successfully tackle had to do with a point noted very early on in research (Brown et al., 1990); the translation of a word may depend on words quite f[40]from it, so the interdependencies of text parts should be accounted. The revolutionary paper of *Attention is All You Need* by Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin (2017) introduced an approach that greatly boosted the efficiency accuracy of machine translation systems by allowing parallel computations and allow

the model to better extract the meaning of input sequences by diverging from the sequential processing of input.

The previous constituted the stepping stone for the beginning of the era of Large Language Models (LLMs), trained on vast corpora and being able to handle diverse tasks with significant performance after limited fine-tuning (Zhu et al., 2024).

Appendix VII

Visualization of Covid Dataset

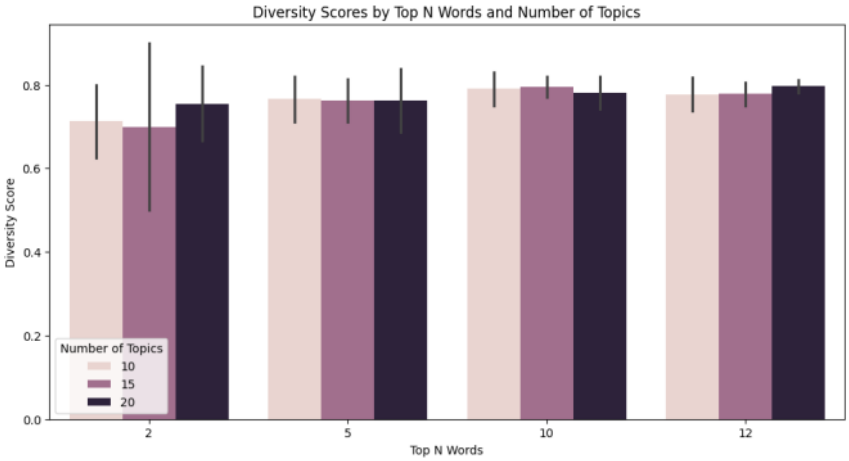

Figure 36: Diversity score as it changes for various top-N words and number of topics.

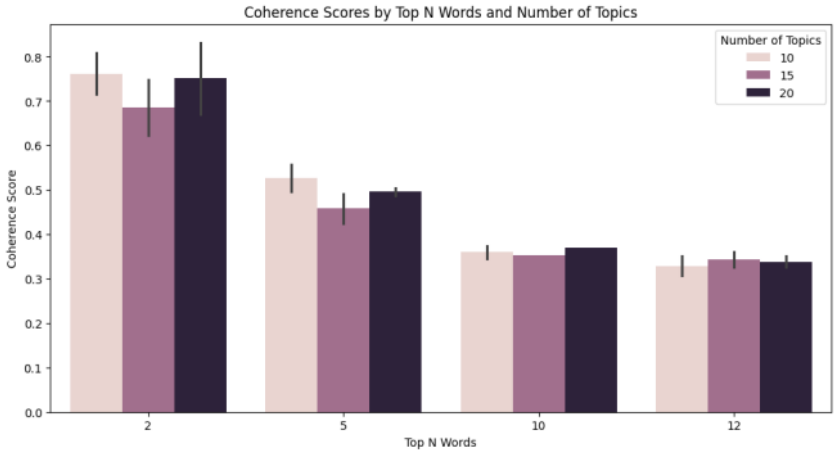

Figure 37: Coherence score as it changes for various top-N words and number of topics.

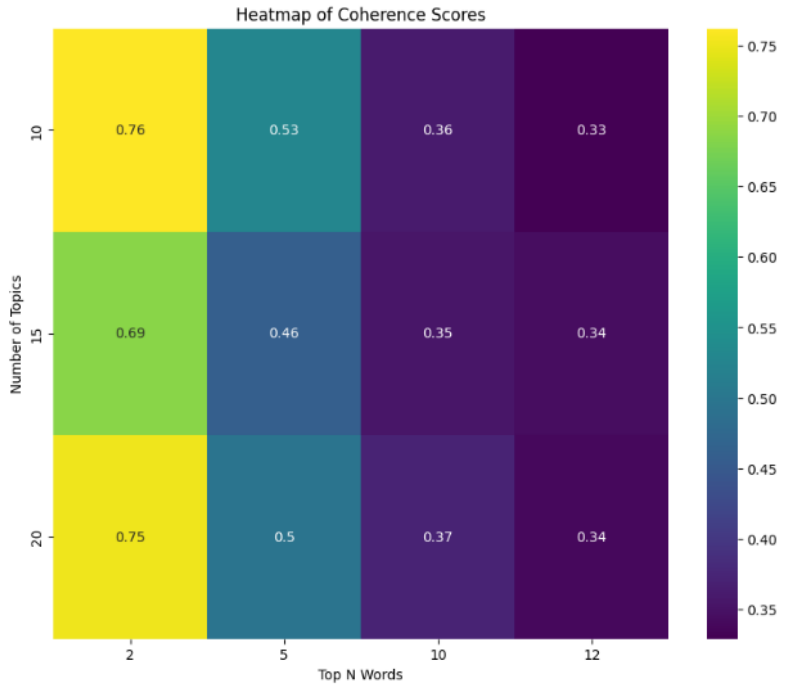

Figure 38: Heat map visualization for Coherence scores for various top-N words and number of topics.

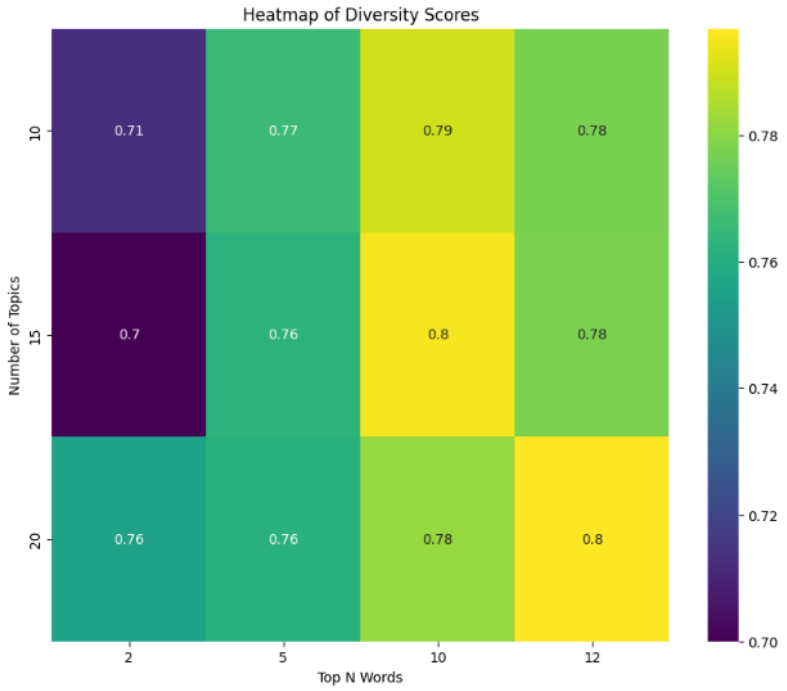

Figure 39: Heat map visualization for Diversity scores for various top-N words and number of topics.

The above analysis focuses on two key evaluation metrics (Diversity and Coherence Score) which are visualized across different numbers of topics (10, 15, and 20) and Top N Words (2, 5, 10, and 12).

The heatmap of Diversity Scores indicates that for 10 topics, the score improves slightly with an increase in Top N Words, maxing out at 0.79 for 10 words and stabilizing around 0.78 for 12 words. Similarly, for 15 topics, the score reaches a maximum of 0.80 with 10 words. Additionally, for 20 topics, consistent performance is observed across different numbers of words, with the highest diversity score (0.80) at 10 and 12 words. The bar plot also shows the highest diversity for 10 words across different topic counts, with small variations.

The heatmap of Coherence Scores reveals that for 10 topics, the highest coherence (0.76) is achieved using 2 words, this value significantly declines as the number of words increases. For 15 topics we have the highest coherence (0.69) with 2 words, following a similar tendency with more words (reduce on the coherence). Moreover, for 20 topics, the highest coherence (0.75) is achieved again with 2 words, with the score decreasing as the number of words increases. From the bar plot is evident that models with fewer words per topic (2 or 5) tend to have better coherence.

In general, optimal diversity is achieved with 10 words per topic across different topic counts, while coherence is best with fewer words per topic (2 or 5). Therefore, probably a model with 15 topics and 5 words could be a reasonable approach to get a balanced diversity and coherence.

Figure 40: World cloud visualization of most common words within the Covid data set.

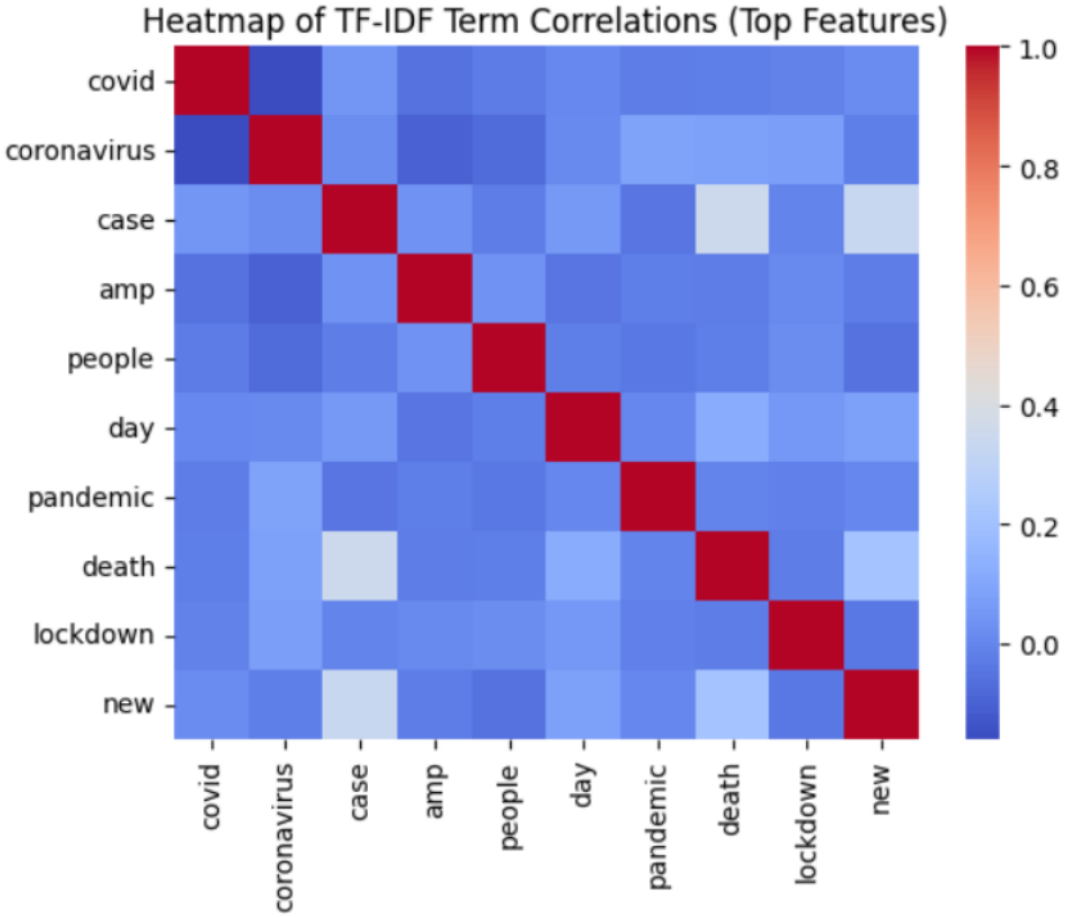

Figure 41: Heat map signifying correlation between common words within the documents in the covid data set.

The TF-IDF term correlation heatmap highlights the relationships between top features based on their co-occurrence in documents. Specific, terms such as "death" and "pandemic" exhibit moderate correlations, this indicates that these terms frequently appear together in discussions pertaining to the COVID-19 context. This observation is to be expected, since the pandemic's death toll was a common topic of discussion. Similarly, terms like "lockdown" and "new" demonstrate significant correlations, reflecting their recurrent appearance in the same document. This signifies discussions surrounding the consecutive and “new” lockdowns imposed during the pandemic.

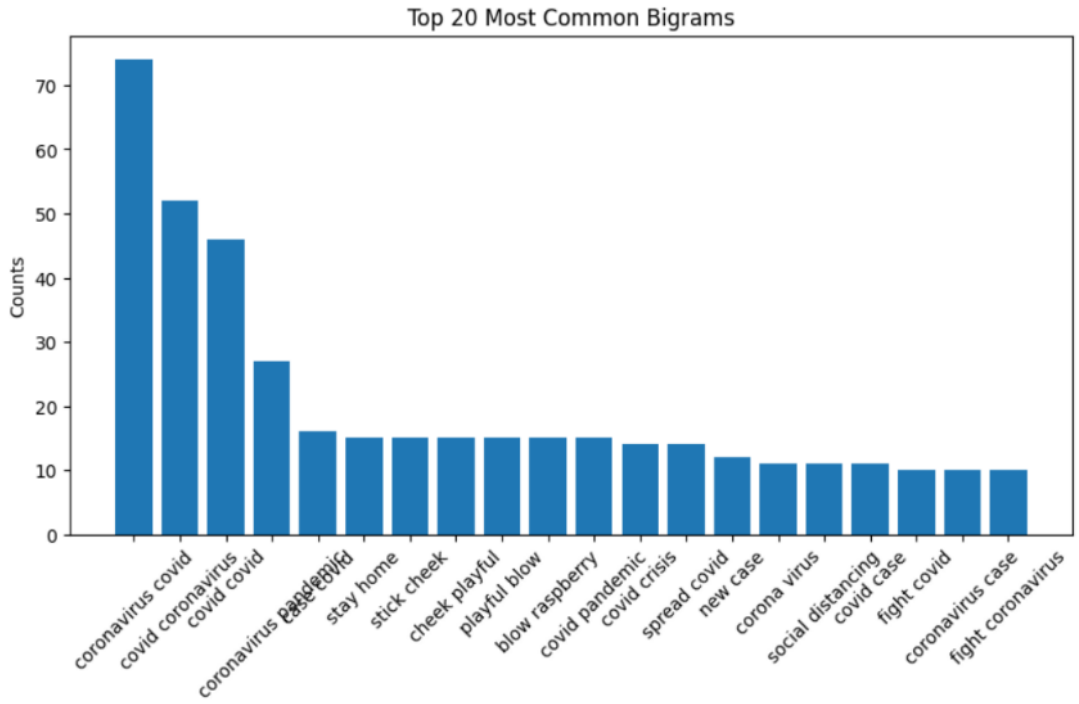

Figure 42: Most common bi-grams in covid dataset.

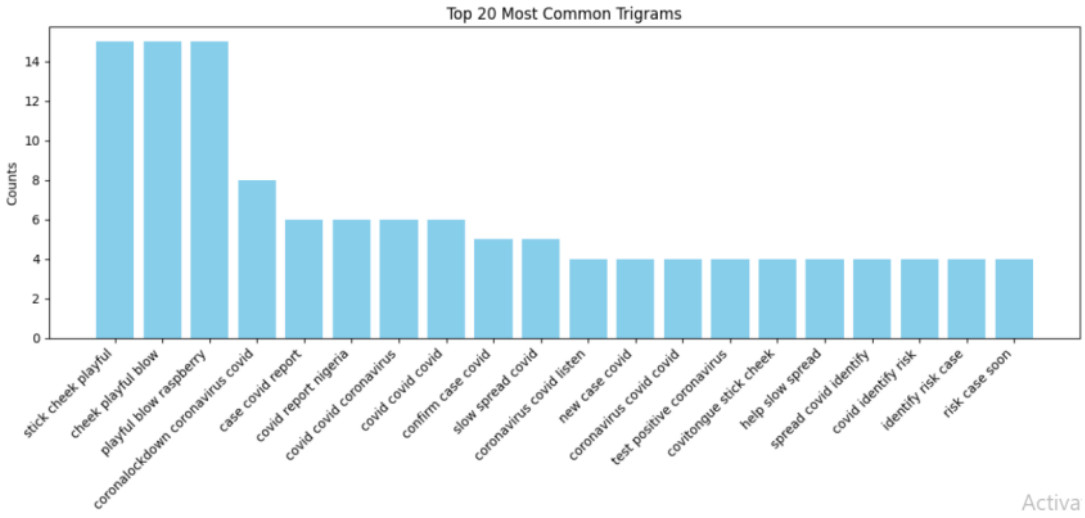

Figure 43: Most common tri-grams in covid dataset.

Visualizations for customer support dataset

Figure 44: World cloud visualization of most common words within the customer support data set.

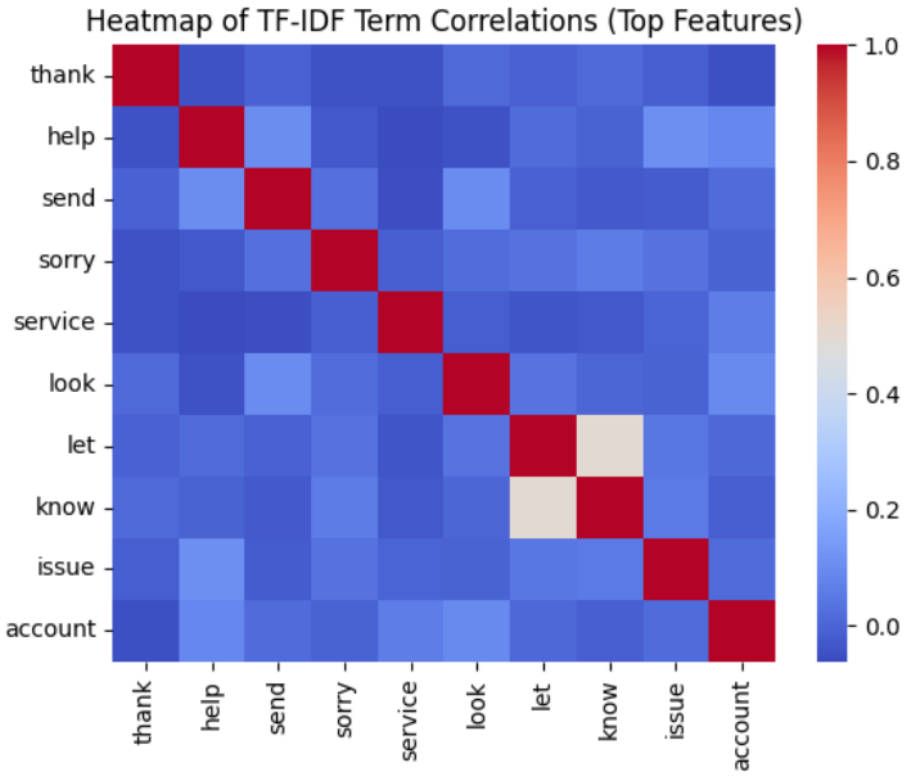

Figure 45: Heat map signifying correlation between common words within the documents for tha customer support dataset.

The heatmap visualizes correlations between top TF-IDF features in a text dataset, with red indicating high correlation and blue indicating low correlation. High term correlations help identify common phrases, which are useful for our use case and NLP tasks in general. As such we see that the diagonal cells show perfect correlations (1.0) of terms with themselves, which is to be expected. Additionally, we observe that "let" and "know" exhibit a high correlation, which possibly suggests frequent use of phrases like "let me know." There is also a *moderate* correlation between "thank" and "help," indicating phrases such as "thank you for your help." "Issue" and "account" also show some degree of correlation, possibly suggesting discussions around account issues. Most other term pairs show low correlations, signifying that these terms do not frequently appear together.

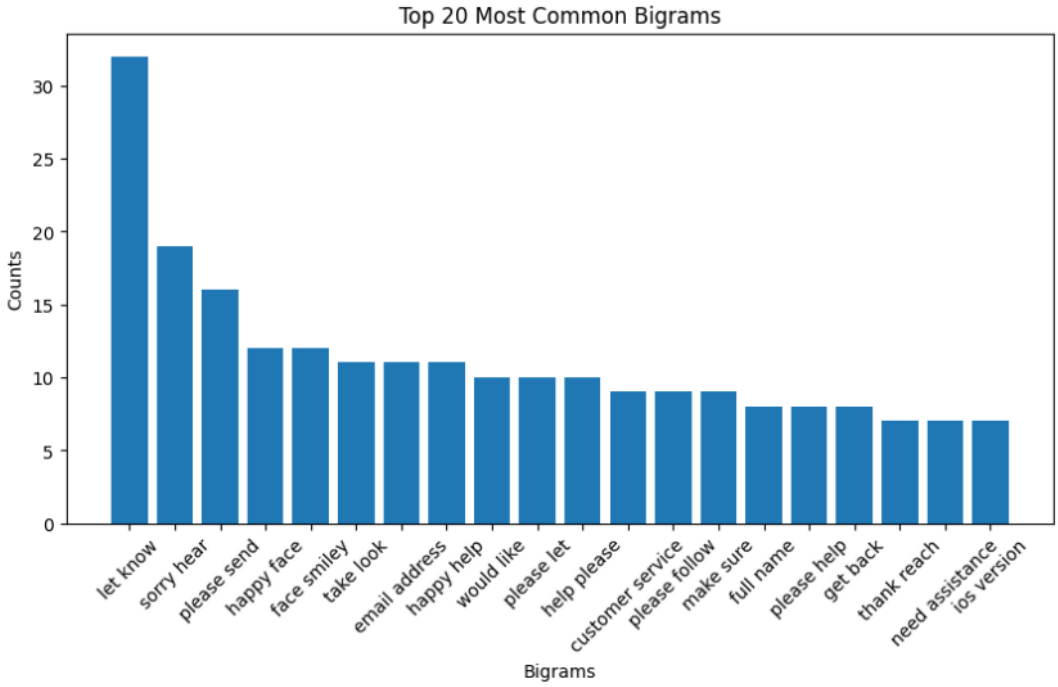

Figure 46: Most common bi-grams for the customer support dataset.

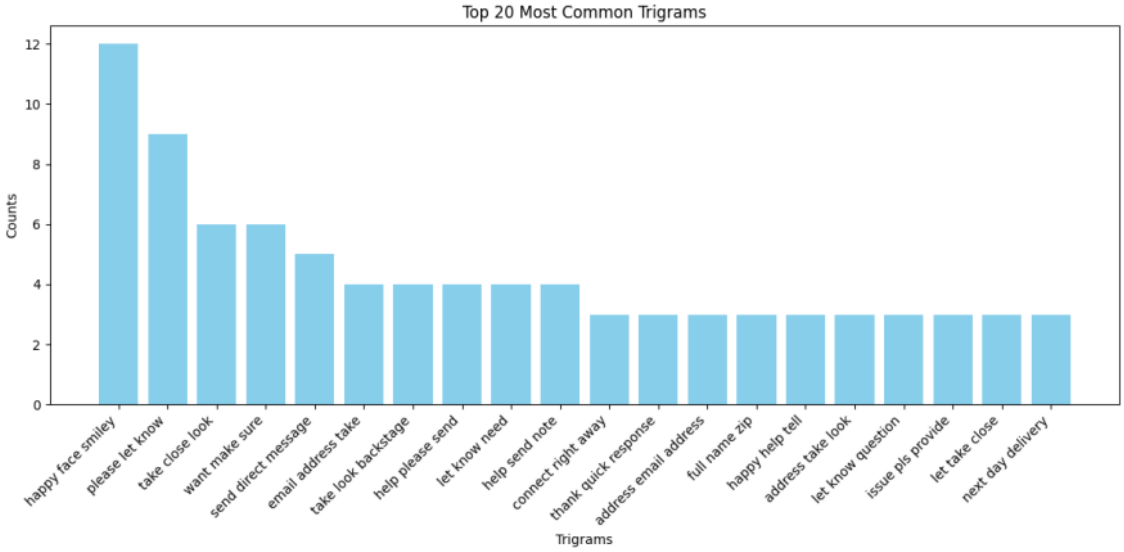

Figure 47: Most common tri-grams for the customer support dataset.

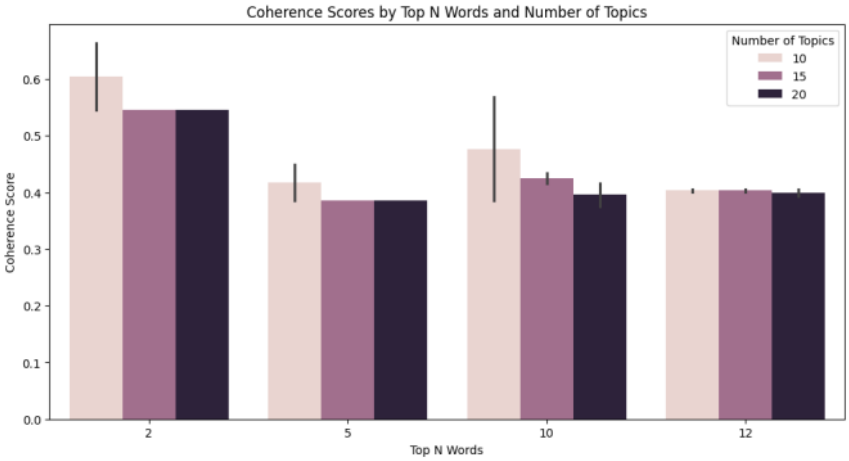

Figure 48: Coherence score as it changes for various top-N words and number of topics.

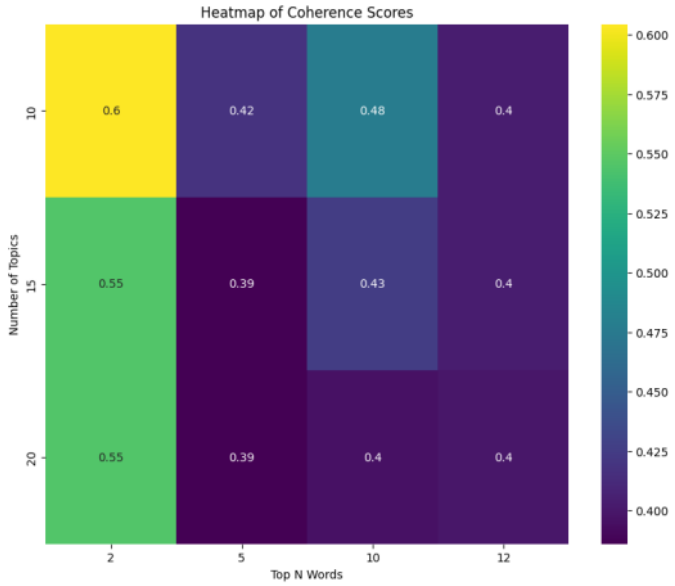

Figure 49: Heat map visualization for Coherence scores for various top-N words and number of topics for the customer support dataset.

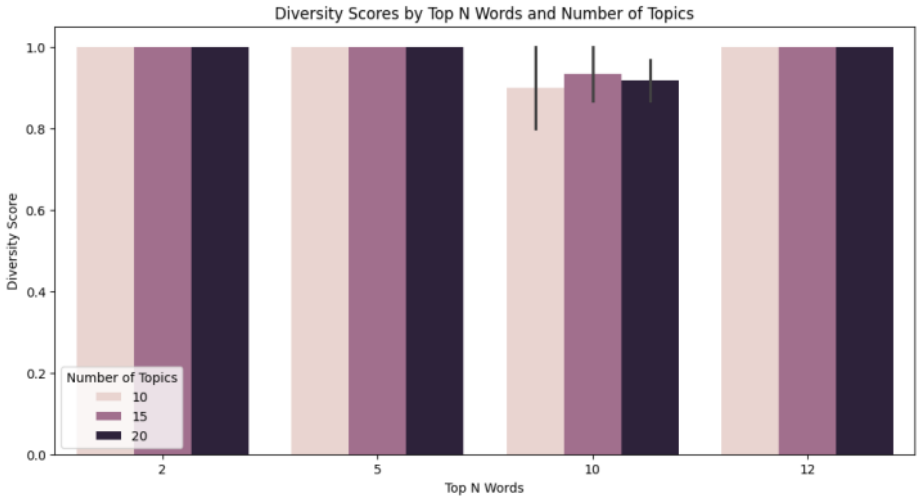

Figure 50: Diversity score as it changes for various top-N words and number of topics.

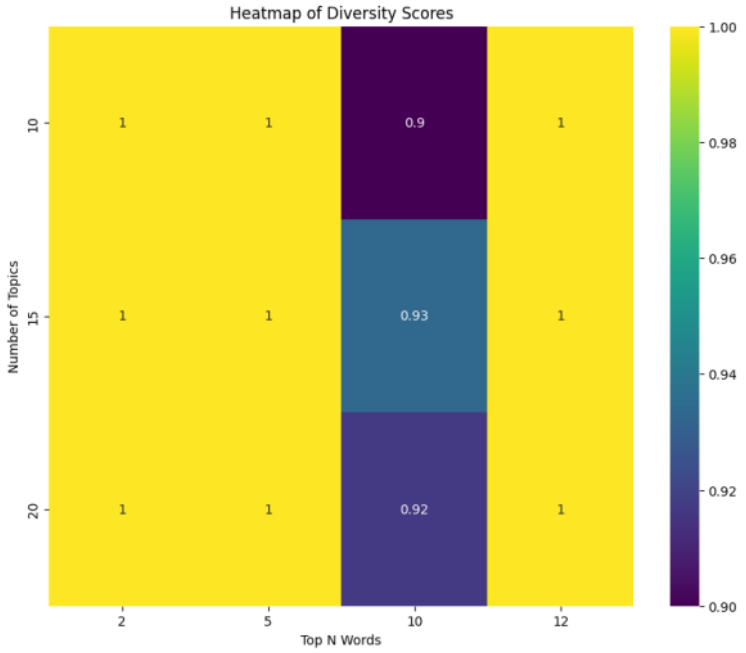

Figure 51: Heat map visualization for Diversity scores for various top-N words and number of topics for the customer support dataset.

The above explanatory chart provides an insight on the evaluation based on varying parameters such as the number of topics (10, 15, and 20) and the top N words used (2, 5, 10, and 12).

The heatmap and bar plot related to the diversity scores (figures x, x) display that for most configurations, the diversity score remains constant at 1. However, using the top 10 words resulted in a slight decrease in diversity scores for 10, 15, and 20 topics, this suggests an overlap in the topics generated with this word count.

The coherence scores, visualized through a heatmap and bar plot (figures (x, x), show more variation. With 10 topics, coherence is highest when using the top 2 words (0.6) and decreases with more words, reaching 0.4 for 12 words. Additionally, for 15 and 20 topics, coherence score starts lower (0.55 for 2 words) and decreases further as the number of words increases. The lowest coherence scores are observed when using the top 5 words across all topic counts.

Overview & Key Insights

The above showcases that a trade-off between diversity and coherence is evident, since high diversity scores are maintained across most settings, but coherence tends to decline as more words or topics are introduced. This balance is crucial for effective topic modeling, since for applications requiring a broad coverage of themes, high diversity is beneficial. On the other hand, in our case where clear/distinct topics are needed, higher coherence should be prioritized and we should focus on fewer, more significant words (and possibly by limiting the number of topics).

Therefore, it is important to understand that while developing topic models, we should balance diversity and coherence based on the use case in hand.

- For broad thematic coverage, prioritize diversity.
- For clear, distinct topics, prioritize coherence.

Summarization

Text summarization is a crucial tool for transforming extensive information into meaningful summaries. Here we used an encoder-decoder architecture, incorporating Long Short-Term Memory (LSTM) layers and an attention mechanism (GeeksforGeeks, 2024; Pai, 2023). As dataset, we used a CSV file (test.csv), containing two columns ('article' and 'highlights'), representing full texts and their corresponding summaries respectively. We only applied tokenization for data preparation. Text and summary sequences are converted to the same length, which is something that was found by examining the 95th percentile lengths. This ensured consistent/common input sizes for the model (Pai, 2023). The vocabulary sizes, used in defining the embedding layers, are calculated for both articles and summaries.

Additionally, the model used here employes an encoder-decoder structure enhanced with an attention mechanism. In summary, the encoder processes the input text, generating vectors and state values, while the decoder utilizes these to generate summaries. Specifically, the encoder consists of an embedding layer followed by an LSTM layer that returns sequences and state values. The decoder integrates an attention layer that aligns decoder outputs with relevant parts of the encoder outputs, focusing on significant parts of the input sequence.

Moreover, training the model involved padded sequences, with the target sequences for training being shifted versions of the summary sequences. The model is trained for three epochs (due to time limitations) with a batch size of 64 and utilizes the *sparse_categorical_crossentropy* loss function, suitable for sequence prediction tasks. During inference, separate encoder and decoder models facilitate step-by-step summary generation, allowing the model to predict one word at a time until a stop condition is met.

Generally, the results are not great since the model offers Incomprehensible summaries with repeating words usually. This was expected due to the training of the model (13,000 instances). If better results were derived, we would have to tune the model and evaluate it using BLEU, ROUGE, or METEOR for quantitative assessment nevertheless in our case and since the summarization here constitutes just something extra, we did not proceed with evaluation of the results.

Bibliography

Ar5iv. (n.d.). *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. https://ar5iv.labs.arxiv.org/html/2203.05794

Ashok, D., & Lipton, Z. C. (2023). PromptNER: Prompting for named entity recognition. arXiv. https://doi.org/10.48550/arXiv.2305.15444

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. arXiv. https://doi.org/10.48550/arXiv.1409.0473

Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media Inc.

Borthwick, A. (1999). *A maximum entropy approach to named entity recognition* (Doctoral dissertation). Computer Science Department, New York University. Retrieved from https://cs.nyu.edu/media/publications/borthwick_andrew.pdf on June 10, 2024.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., & Roossin, P. S. (1990). A Statistical Approach to Machine Translation. *Computational Linguistics*, *16*(2), 79–85.

Brownlee, J. (2020, August 27). Tune hyperparameters for Classification Machine Learning Algorithms. MachineLearningMastery.com. https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/

Cho, H. C., Okazaki, N., Miwa, M., & Tsujii, J. (2013). Named entity recognition with multiple segment representations. Information Processing & Management, 49(4), 954-965. Retrieved from https://www.academia.edu/12852833/Named_entity_recognition_with_multiple_segment_representations on July 18, 2024

Chollet, F. (2017, September 29). A ten-minute introduction to sequence-to-sequence learning in Keras. Keras Blog. https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

Dasgupta, S. (2021, September 16). *Latent Semantic Analysis and its Uses in Natural Language Processing*. Analytics Vidhya. Retrieved July 12, 2024, https://www.analyticsvidhya.com/blog/2021/09/latent-semantic-analysis-and-its-uses-in-natural-language-processing/

DeepL. (n.d.). *DeepL API*. Retrieved June 25, 2024, from https://www.deepl.com/pro-api

Explosion. (n.d.). *spaCy: Industrial-strength natural language processing in Python*. Retrieved June 1, 2024, from https://spacy.io

Gallo, I., Binaghi, E., Carullo, M., & Lamberti, N. (2008). Named entity recognition by neural sliding window. In Proceedings of the Document Analysis Systems (pp. 567-573). IEEE. https://doi.org/10.1109/DAS.2008.85

GeeksforGeeks. (2024, June 10). *What is LSTM Long Short Term Memory?* GeeksforGeeks. Retrieved July 19, 2024, https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

Google BigQuery. (n.d.). BigQuery: Cloud data warehouse. Google Cloud. Retrieved July 20, 2024, from https://cloud.google.com/bigquery

Google. (n.d.). *Google Colaboratory*. Retrieved June 1, 2024, from https://colab.research.google.com/

Hutchins, J. (2004). *Two precursors of machine translation: Artsrouni and Trojanskij* (p. 14). https://web.archive.org/web/20181126102920/http://www.hutchinsweb.me.uk/IJT-2004.pdf

Keras. (n.d.). Probabilistic losses. Keras API. Retrieved from https://keras.io/api/losses/probabilistic_losses/#sparsecategoricalcrossentropy-class

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (NAACL 2016). arXiv. https://doi.org/10.48550/arXiv.1603.01360

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 260-270). Association for Computational Linguistics. https://doi.org/10.48550/arXiv.1603.01360

Lotfy, M. (n.d.). WMT 2014 English-German: German to English machine translation dataset. Kaggle. Retrieved June 1, 2024, from https://www.kaggle.com/datasets/mohamedlotfy50/wmt-2014-english-german

Luiggi, T., Soulier, L., Guigue, V., Jendoubi, S., & Baelde, A. (2023). Dynamic named entity recognition. *Proceedings of the 2023 ACM Symposium on Applied Computing*, 8 pages. arXiv. https://doi.org/10.48550/arXiv.2302.10314

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1412-1421). arXiv. https://doi.org/10.48550/arXiv.1508.04025

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 51-56).

Miller, G. A. (1995). WordNet: A lexical database for English. Communications of the ACM, 38(11), 39-41. https://doi.org/10.1145/219717.219748

OpenAI. (2021). InstructGPT. OpenAI. https://beta.openai.com/docs/guides/instructGPT

Oxbridge Editing. (2024, March 1). What are the 14 punctuation marks? Oxbridge Editing. https://www.oxbridgeediting.co.uk/blog/what-are-the-14-punctuation-marks/

Pai, A. (2023, December 26). *Comprehensive Guide to Text Summarization using Deep Learning in Python*. Analytics Vidhya. Retrieved July 18, 2024, https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/

Pakhale, K. (2023). Comprehensive overview of named entity recognition: Models, domain-specific applications and challenges. arXiv. https://doi.org/10.48550/arXiv.2309.14084

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311-318). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830. Retrieved from https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

Python Software Foundation. (n.d.). *re — Regular expression operations*. In Python 3.9.13 documentation. Retrieved June 1, 2024, from https://docs.python.org/3/library/re.html

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv. https://doi.org/10.48550/arXiv.1910.10683

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv. https://doi.org/10.48550/arXiv.1910.01108

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. arXiv. https://arxiv.org/abs/1409.3215

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv. https://doi.org/10.48550/arXiv.1706.03762

Zhu, W., Liu, H., Dong, Q., Xu, J., Huang, S., Kong, L., Chen, J., & Li, L. (2024). Multilingual machine translation with large language models: Empirical results and analysis. arXiv. https://doi.org/10.48550/arXiv.2304.04675

ITC6110_NLP_Report_Kapsalis_Stavrogiannis_Tzoras_Final.d...

ORIGINALITY REPORT

12%	10%	9%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Mohanad Abukmeil, Stefano Ferrari, Angelo Genovese, Vincenzo Piuri, Fabio Scotti. "A Survey of Unsupervised Generative Models for Exploratory Data Analysis and Representation Learning", ACM Computing Surveys, 2022 Publication	1%
2	link.springer.com Internet Source	1%
3	Submitted to Korea University Student Paper	1%
4	Submitted to Allen D. Nease High School Student Paper	<1%
5	Sudeshna Das, Jiaul H Paik. "Context-sensitive gender inference of named entities in text", Information Processing & Management, 2021 Publication	<1%
6	Submitted to The Open University of Hong Kong Student Paper	<1%

7	www.riverpublishers.com Internet Source	<1 %
8	pdfs.semanticscholar.org Internet Source	<1 %
9	www.psychologie-aktuell.com Internet Source	<1 %
10	nclt.dcu.ie Internet Source	<1 %
11	yaginu.hatenablog.com Internet Source	<1 %
12	ouci.dntb.gov.ua Internet Source	<1 %
13	idmc.univ-lorraine.fr Internet Source	<1 %
14	openportal.isti.cnr.it Internet Source	<1 %
15	Submitted to University of Hong Kong Student Paper	<1 %
16	Submitted to IUBH - Internationale Hochschule Bad Honnef-Bonn Student Paper	<1 %
17	arxiv.org Internet Source	<1 %

18	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
19	www.aclweb.org Internet Source	<1 %
20	journals.plos.org Internet Source	<1 %
21	jurnal.polibatam.ac.id Internet Source	<1 %
22	Submitted to Georgia Institute of Technology Main Campus Student Paper	<1 %
23	Submitted to University of Ulster Student Paper	<1 %
24	atm.amegroups.com Internet Source	<1 %
25	export.arxiv.org Internet Source	<1 %
26	Submitted to Charles University Student Paper	<1 %
27	Submitted to National University of Ireland, Galway Student Paper	<1 %
28	bmcbioinformatics.biomedcentral.com Internet Source	<1 %

29	github.com Internet Source	<1%
30	sakshi-nkulkarni.medium.com Internet Source	<1%
31	theses.hal.science Internet Source	<1%
32	www.springerprofessional.de Internet Source	<1%
33	Submitted to Australian National University Student Paper	<1%
34	Submitted to Napier University Student Paper	<1%
35	Submitted to Taylor's Education Group Student Paper	<1%
36	Submitted to University of Maryland, University College Student Paper	<1%
37	digitalscholarship.unlv.edu Internet Source	<1%
38	taln2017.cnrs.fr Internet Source	<1%
39	www.ncbi.nlm.nih.gov Internet Source	<1%
40	Submitted to University of Sunderland Student Paper	

		<1%
41	Submitted to University of Southampton Student Paper	<1%
42	curis.ku.dk Internet Source	<1%
43	web.archive.org Internet Source	<1%
44	www.arxiv-vanity.com Internet Source	<1%
45	Submitted to Chiang Mai University Student Paper	<1%
46	Ruichuan Zhang, Nora El-Gohary. "Transformer-based approach for automated context-aware IFC-regulation semantic information alignment", Automation in Construction, 2023 Publication	<1%
47	Submitted to Hong Kong University of Science and Technology Student Paper	<1%
48	Submitted to Ikon Institute Student Paper	<1%
49	Mariana Rodrigues Makiuchi, Tifani Warnita, Kuniaki Uto, Koichi Shinoda. "Multimodal	<1%

Fusion of BERT-CNN and Gated CNN Representations for Depression Detection", Proceedings of the 9th International on Audio/Visual Emotion Challenge and Workshop - AVEC '19, 2019
Publication

50	ojs.aaai.org Internet Source	<1 %
51	docslib.org Internet Source	<1 %
52	publikationen.sulb.uni-saarland.de Internet Source	<1 %
53	scholarworks.montana.edu Internet Source	<1 %
54	Submitted to Monash University Student Paper	<1 %
55	bmcoralhealth.biomedcentral.com Internet Source	<1 %
56	board.coveredca.com Internet Source	<1 %
57	eda.europa.eu Internet Source	<1 %
58	www.mdpi.com Internet Source	<1 %
59	www.packtpub.com	

	Internet Source	<1%
60	Gizem Aras, Didem Makaroğlu, Seniz Demir, Altan Cakir. "An evaluation of recent neural sequence tagging models in Turkish named entity recognition", Expert Systems with Applications, 2021 Publication	<1%
61	aut.researchgateway.ac.nz Internet Source	<1%
62	ojs.bbwpublisher.com Internet Source	<1%
63	pure.rug.nl Internet Source	<1%
64	reunir.unir.net Internet Source	<1%
65	searchengineland.com Internet Source	<1%
66	www.frontiersin.org Internet Source	<1%
67	Viktar Atliha. "Improving image captioning methods using machine learning approaches", Vilnius Gediminas Technical University, 2023 Publication	<1%

68	Wang, Chengfei. "User Feedback Analysis for Business Intelligence: Semantics Sentiment and Model Robustness", Auburn University, 2024 Publication	<1 %
69	aclanthology.org Internet Source	<1 %
70	alvinntnu.github.io Internet Source	<1 %
71	amtaweb.org Internet Source	<1 %
72	bora.uib.no Internet Source	<1 %
73	commons.stmarytx.edu Internet Source	<1 %
74	repositorio.ufmg.br Internet Source	<1 %
75	research-explorer.ista.ac.at Internet Source	<1 %
76	scholarshare.temple.edu Internet Source	<1 %
77	www.iaarc.org Internet Source	<1 %

78	"Advances in Data and Information Sciences", Springer Science and Business Media LLC, 2022 Publication	<1 %
79	"ECAI 2020", IOS Press, 2020 Publication	<1 %
80	Majumdar, Sayantan. "Groundwater Withdrawal Estimation Using Integrated Remote Sensing Products and Machine Learning", Missouri University of Science and Technology, 2023 Publication	<1 %
81	Stanojevic, Marija. "Domain Adaptation Applications to Complex High-Dimensional Target Data", Temple University, 2023 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		