ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ + ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

ΑΘΗΝΑ
Έρευνα & Καινοτομία
Τεχνολογίες Πληροφορίας

ΙΕΛ
ILSP

M908 NLP (Summer Semester 2023)
Mini Project: Sarcasm Detection

Report

Kleopatra Karapanagiotou (lt12200010)
Giorgos Kordonis (lt12200028)
Sofia Tsenekidou (lt12200023)
Eleni Mpatsi(lt12200015)

Contents

Introduction

Problem Statement

Recognizing sarcasm is a challenge for almost 50% of people. This problem becomes even more pronounced in the context of news headlines, where sarcasm can have significant implications for public opinion and political discourse. Sarcastic comments generally tend to report false information with an intent to invert the sentiment of the expression that they seek to report.
Such an inversion of sentiment could be misleading to the unwary people, who follow such news posts at face value. Specifically, in the context of NLP, sarcasm detection is a specific case of sentiment analysis, where instead of detecting a sentiment in the whole spectrum, the focus is on sarcasm. Therefore, the task of this field is to detect if a given text is sarcastic or not. Nevertheless, the fact that we cannot simply use a set of reference words without prior context to detect sarcasm, makes this classification task complicated.

Project Description

In this project we aim at detecting news headlines of Greek news websites that have been written with sarcastic intent and hence misrepresent actual facts about current affairs in the country. For this purpose, we generated a new dataset, including sarcastic and non-sarcastic headlines of a specific time period and experimented with training traditional classifiers, as well as neural networks to examine their performance. Since our classifier aims at predicting headlines in Greek, the Greek BERT model was chosen as the most appropriate embedding representation in the neural networks. Dataset analysis and model results are discussed in the following sections.

Structure

The attached.zip file contains 4 files named “Phases”, which should be run in the respective order:

Phase_1: This folder includes the Scrapy project files. Scrapy needs to be installed for the code to run, and as soon as the library is installed, a Scrapy project has been created, along with the spider.py files found in the “spiders” folder.

Phase_2_ This folder includes the **mini_project_tsenekidou.ipynb file** which contains the code of preprocessing and cleaning the raw data. It also contains the **FullDataset_Koulouri_Vatraxi (1).json** file which contains the sarcastic data from the web scraping process. If you want to run the code you will need the Kaggle dataset as well which contains the non-sarcastic data before submerging them into a single dataset. Due to capacity reasons, this file is not included in this folder but the **link** is provided to load it in your computer. [https://www.kaggle.com/datasets/kpittos/news-articles.](https://www.kaggle.com/datasets/kpittos/news-articles)

Finally, this folder includes a **data.csv** file that contains the final dataset after the cleaning.

Phase_3 : This folder contains a main.py file, which includes certain tasks. Firstly, the raw data is loaded from two different files (raw_sarc.csv and raw_news.csv) and a dataframe is created by concatenating the two sets and shuffling them. Then, after loading the appropriate libraries (Transformers, Tensorflow), Greek Bert is implemented. We achieve the implementation by loading the tokenizer and the model and creating a function for extracting the sentence embeddings from the main dataframe with our data. After extracting the sentence embeddings, the first MLP model is executed, which has as an input the embeddings from the downsampled data. Afterwards, we use a small script for getting the sentence embeddings after we augment our data, in order to check if the model can perform better with this specific input. This augmentation and relative embeddings extraction task is separated into two parts, since its completion required a high computational load. As a result, we chose to separate the augmented dataframe into 4 subsets, and then concatenate them. After this, the second MLP model is trained with the augmented data. Next, some snippets of code were used for gaining insight for our raw downsampled data, specifically with cosine similarity and clustering from KMeans. Finally, a PCA algorithm is implemented, in order to visualize the embeddings into groups, after reducing their dimensions. Also, the file includes the two saved weights of the models, two figures that show the history of the training and validation of the MLP, two figures that show the confusion matrices of the two models and two figures of the visualization of the groups from the PCA.

Phase_4: Contains 2 subfiles, including the experiment scripts (biLSTM_BiGRU_Attention, BERT_Finetuning) and results in figures of the **Undersampled** and **Augmented** dataset respectively. Furthermore, the file “Augmented” includes an “Application.ipynb " file, which contains the relevant script, in order to run the Sarcasm Detector in a gradio interface. This code generates an expirable, public URL, redirecting to the Sarcasm Detector app, trained on the fine-tuned BERT.

Data

Background

Our dataset was created based on the rationale of Misra, Rishabh and PrahalArora in their work "Sarcasm Detection using News Headlines Dataset." AI Open (2023). We merged news headlines of Greek from the following Kaggle dataset: https://www.kaggle.com/datasets/kpittos/news-articles with data that we generated through crawling and scraping from two pure sarcastic websites in Greece : https://www.tokoulouri.com/ and https://tovatraxi.com/. Since news headlines are written by professionals in a formal manner, we assume, there are no spelling mistakes and informal usage. This reduces the sparsity and also increases the chance of finding pre-trained embeddings. Furthermore, since the sole purpose of “To Koulouri” and “To vatrahi” is to publish sarcastic news, we get high-quality labels with much less noise as compared to Twitter datasets. Unlike tweets which are replies to other tweets, the news headlines we obtained are self-contained. This would help us in teasing apart the real sarcastic elements.

Dataset Generation

In this part of the report, we will discuss the process and rationale behind using Scrapy, a powerful Python library for web scraping, to extract article titles from websites “To Koulouri” and “To Vatraxi”. The extracted titles will serve as the training data for a machine learning model that aims to differentiate between sarcastic and non-sarcastic speech. Sarcasm detection has become a significant challenge in natural language processing. The ability to automatically identify sarcastic statements can have applications in sentiment analysis, social media monitoring, and even improving conversational AI systems. To address this challenge, we decided to leverage web scraping techniques and machine learning to build a classifier capable of identifying sarcastic speech.

Scrapy: a Web Scraping Framework: Scrapy provides a robust framework for web scraping tasks, offering features such as efficient request handling, asynchronous processing, and powerful parsing capabilities. These features make Scrapy an ideal choice for extracting large amounts of data from websites in a structured and efficient manner.

To Koulouri/To Vatraxi: Sarcastic News Satire Platforms: “To Koulouri” and “To Vatraxi” are two well-known Greek news satire platforms that publish articles with a sarcastic tone. The websites’ headline titles serve as a valuable resource for training a machine learning model to recognize sarcasm in text.

Extracting Article Titles using Scrapy: We utilized Scrapy to crawl “To Koulouri” and “To Vatraxi” websites and extract article titles from various sections. By defining rules within Scrapy, we directed the spiders to navigate through the websites and locate the relevant elements containing the article titles. Scrapy's CSS selectors proved effective in accurately identifying and extracting the required data.

Here are the two spiders employed for scraping “To Koulouri” and “To Vatraxi” websites, as well as the relevant modifications made to the settings.py file:

The CrawlingSpider class is derived from CrawlSpider, a subclass of scrapy.Spider specifically designed for crawling websites. Our first spider is named 'peterparker', and the allowed_domains attribute is set to 'tokoulouri.com'. The starting URL is 'https://www.tokoulouri.com/'.The rules tuple contains a single rule using LinkExtractor to extract links with one of the following paths: (="politics/", ="international/","science/", “economy/","culture/", "sports/", "opinions/" and "society/"). The callback function, parse_item, is specified to handle the response from the extracted links.

The parse_item function extracts the article title and date using CSS selectors and yields a dictionary with the extracted data.

Similar to the previous spider, the CrawlingSpider class is derived from CrawlSpider. This spider is named 'milesmorales', and the allowed_domains attribute is set to 'tovatraxi.com'. The starting URL is 'https://tovatraxi.com/'.

The rules tuple consists of multiple rules, each using LinkExtractor to extract links from the different sections of the website ("politics", "social", "world", "science", art", "entertainment", "specials" and "business"). The callback function, parse_item, is used to handle the response from the extracted links.

The parse_item function extracts the article title and date using CSS selectors and yields a dictionary with the extracted data.

Finally, the process is executed using the following command line prompts:

1. scrapy crawl peterparker -o output_final.json
2. scrapy crawl 'milesmorales' -o output_frog.json

Data Preprocessing and Cleaning

Data preprocessing and cleaning are crucial steps to ensure the quality and suitability of the data for further analysis and modeling. In our project we followed the steps below.

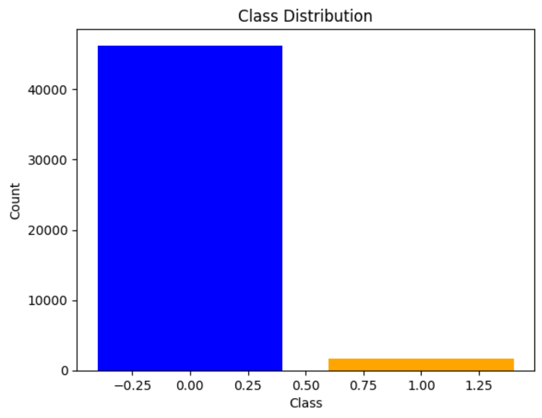
Data Collection: We gathered the raw data from web scraping, and we loaded them in a pandas Dataframe in order to perform an initial exploration of the data to understand its structure, format, and any potential issues that needed to be addressed during cleaning.

Removing HTML Tags: We removed HTML tags from the text.
Removing Irrelevant or Redundant Columns: We removed columns that are irrelevant to our analysis or contain redundant information. In each case we took into account the project's objectives and the relevance of each column to ensure a streamlined and meaningful dataset. We removed the article columns in the Kaggle dataset as we needed only the headlines for our project.
Cleaning: We applied various text cleaning techniques to remove noise and irrelevant information from the text data. Our steps followed the techniques above:
Duplicates Removal: Removing duplicates ensures that each data point is unique and avoids any repetition or redundancy**.**
Handling Missing Data: Removing NaN values ensures that the data used for analysis is complete and reliable, avoiding biased results.
Lowercasing: We converted all text to lowercase to ensure case consistency.
Converting the dates to the desired format: The dates is a crucial part of our analysis since by definition sarcasm is the opposite of what we consider real and to give the side of common reality to our sarcasm detection dataset and thus in our models we decided that there should be an absolute temporal correlation in our data points. Hence, in the final dataset we kept only the headlines from the years 2020, 2021 and 2022 for each class. By including an absolute temporal correlation in our data points, you are aiming to capture the relationship between the headlines and the events happening in the corresponding years. Our approach assumes that headlines from the same years are more likely to refer to similar events or share a common context. This temporal correlation can help provide a stronger foundation for our sarcasm detection analysis. This also can potentially enhance model performance as the models learn patterns and sarcasm indicators specific to those years, which may be more reliable and accurate for the targeted temporal period. Finally we did this for linguistic reasons as we suppose that restricting the dataset to specific years, we account for potential shifts in language usage, humor styles, and contextual factors that are relevant to sarcasm detection. This ensures that our models are trained on data that is more representative of the linguistic and cultural landscape during those years.
Filtering the dataset to include headlines only from the years 2020, 2021, and 2022: We do that in order to establish a consistent context within each class as we explain above.
Stopword Removal: We removed commonly used words that contribute little to the overall meaning of a headline. These words, known as stopwords, typically include grammatical words like articles (e.g., "ο," "η"), pronouns (e.g., "αυτός," "το,"), prepositions (e.g., "στο," "στην,"), and conjunctions (e.g., "και," "αλλά," "'η"). In our analysis in which we wish to identify whether the existence and identification of sarcasm is related to a specific choice of words or sarcastic vocabulary that possibly helps our models, stopwords often dominate the frequency distribution but carry little semantic value. By filtering out these insignificant words, we believe that we can focus on the content-bearing terms that convey the primary message or convey important contextual information.

Punctuation Removal: We eliminated punctuation marks from the text. We were curious if punctuation marks play an important role in our sarcasm classification models by providing important information, so we built a baseline model to test this hypothesis and have created 2 versions of our dataset one including the punctuation and one without. More precisely we trained a Support Vector Machine model for both our dataset versions -punctuation cleaned and not - and taking into account the accuracy of both cases we found out that punctuation dataset did not achieve a better score in comparison with the punctuation cleaned version so we suppose that punctuation does not offer anything in sarcasm classification tasks.
Removing English Words and Non-Alphanumeric Characters: We removed them because they do not provide significant semantic or contextual information in the basis of our project. Removing them can simplify the text and focus on the essential linguistic content, making sarcastic speech analysis more accurate and efficient. Also removing them helps extract more meaningful features and improve the quality of the numerical representations such as TF-IDF representation which we will do next.
Data Transformation: We converted the preprocessed text data into suitable formats for modeling, such as numerical representations like TF-IDF or word embeddings.
Data Splitting: Split the preprocessed data into training, validation, and testing sets, ensuring that they are representative of the overall dataset

Data imbalance handling

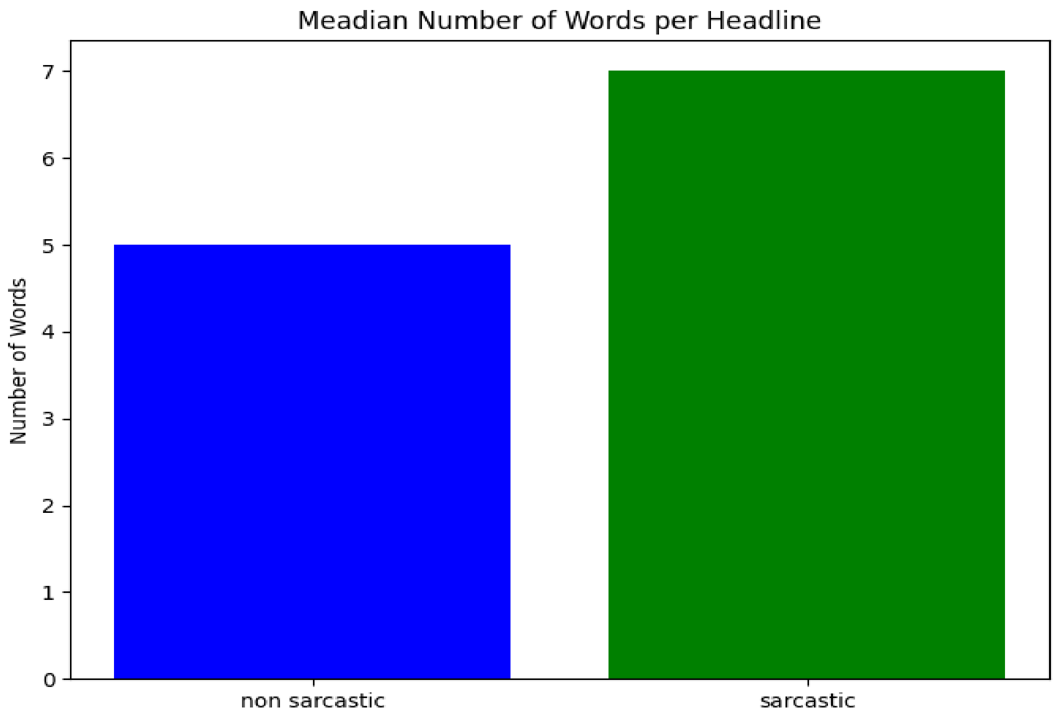
In our attempt to create a whole dataset from the scratch after the process of scraping we had to address the obvious imbalance in our dataset with the class of non-sarcastic headlines being the majority class whether the sarcastic class having significantly smaller proportion of our dataset as shown in the diagram below. To avoid misrepresentation of the sarcastic class, we applied the technique of random undersampling the majority class in order to achieve a balanced dataset.



Data Analysis and Visualisation

In this project, we undertake a comprehensive analysis of textual data,employing text analysis techniques and visualizations to extract valuable insights an uncover patterns within the dataset, specifically focusing on our task of sarcasm detection.

We started our analysis by plotting the median number of words per class in order to investigate whether there is a significant difference of the number of words of each class.



As we see there is not a significant difference in the median number of words between classes, so we assume that text length may not be a distinguishing factor for sarcasm detection in our dataset. Therefore, it is reasonable to assume that the presence or absence of sarcasm is not strongly correlated with the length of the text. This implies that sarcasm can be expressed effectively in both shorter and longer texts, and its detection should rely on other linguistic cues or contextual features. As a disclaimer, we do not necessarily imply that text length is irrelevant for sarcasm detection. It could mean that our dataset of news headlines does not exhibit clear differences in text length between sarcastic and non-sarcastic instances. It is possible that in other datasets or contexts, text length may play a more prominent role.

Visualizing Word Frequency: An Overview of Sarcasm through words

Wordcloud Visualization

Creating word cloud visualizations of the most common words for each class can help examine the existence and identification of words that are more prevalent in sarcastic contexts, why not even create a dictionary of sarcastic words. By analyzing the word cloud representations, patterns and trends can be observed, revealing specific terms associated with sarcasm.The relative size and frequency of words in the word clouds provide insights into their association with sarcasm. Overall, we want to examine whether a qualitative overview of word frequency assists in identifying sarcastic patterns and topics and contributing valuable insights on sarcasm detection.

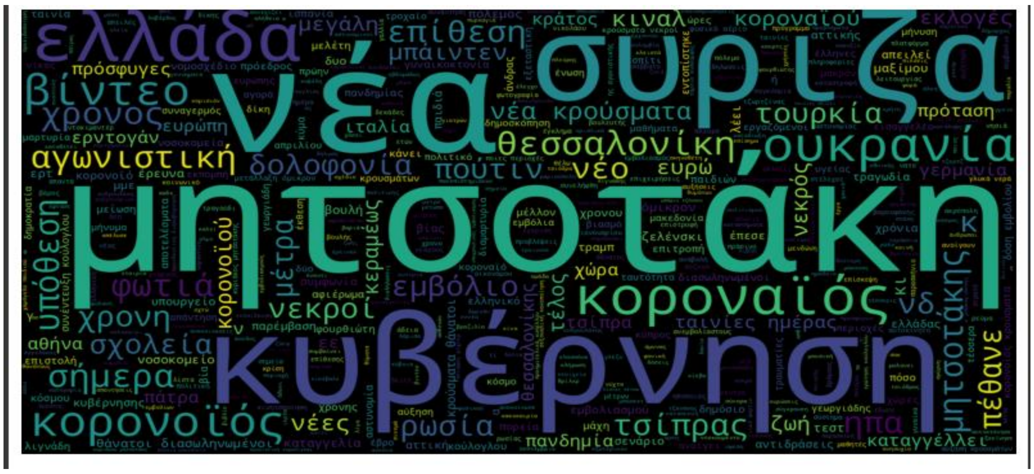


Figure 1: Non sarcastic headlines

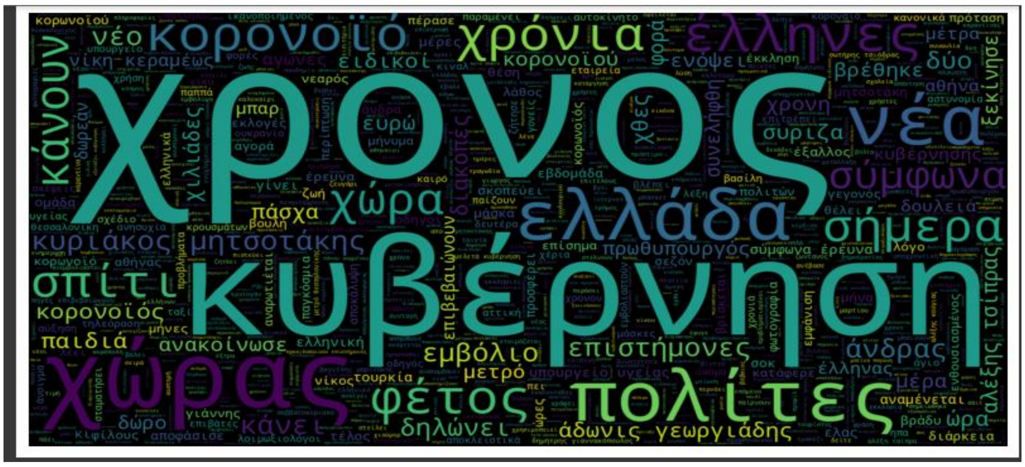
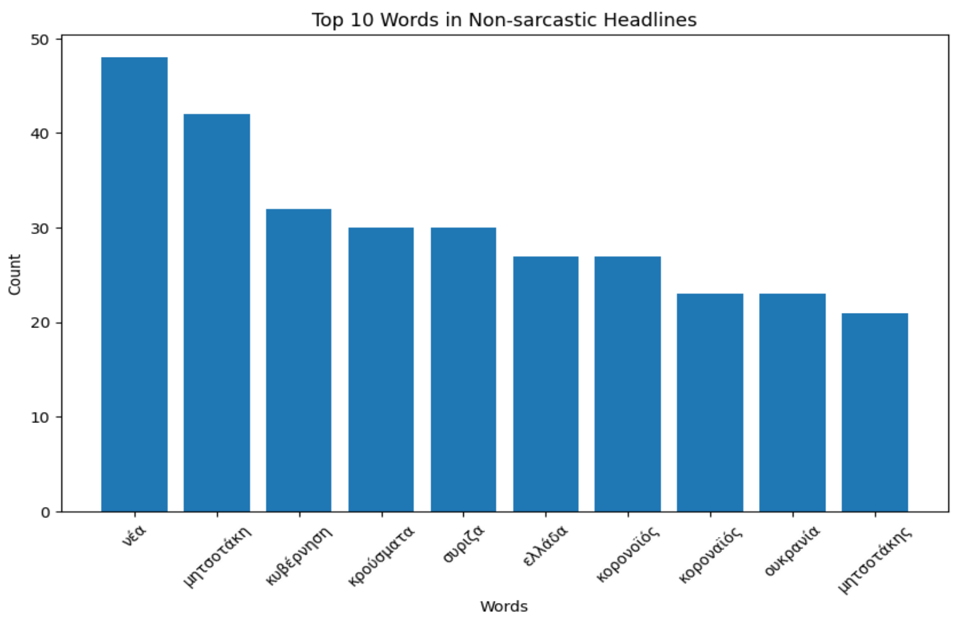
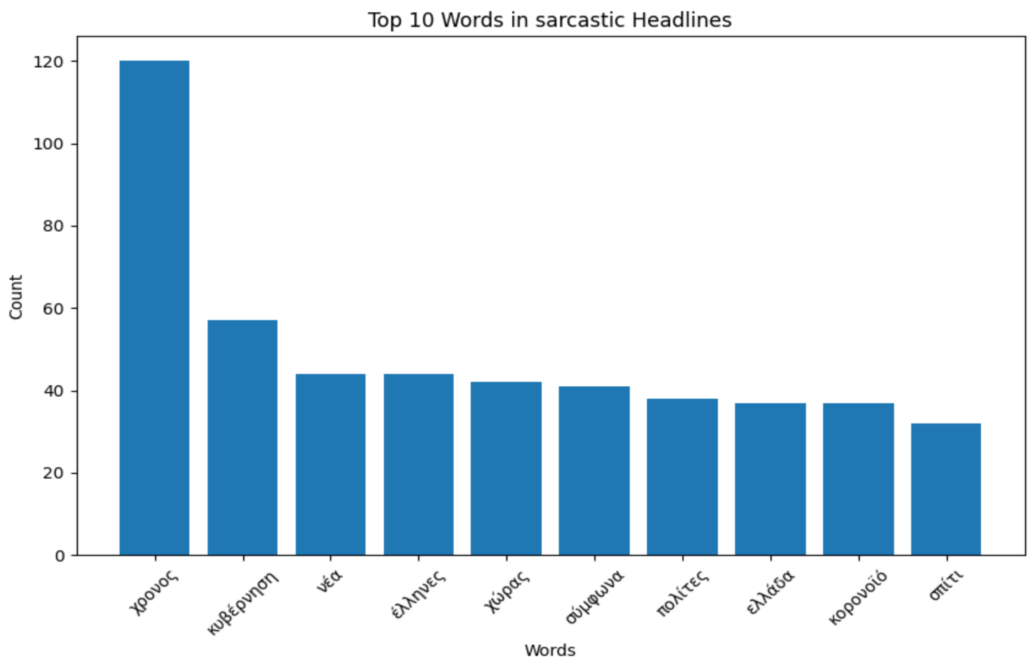


Figure 2: Sarcastic headlines

Bar plot





Based on the above visualizations we can assume that there might not be clear linguistic patterns or distinct vocabulary associated with sarcasm in our dataset. The presence of sarcasm might not be primarily reflected in the frequency of specific words, but rather in other linguistic cues, such as the syntactic structure. We suppose that analyzing the dataset's contextual information or conducting a deeper analysis of the surrounding text may provide more insights into sarcasm detection. Due to the lack of distinct differences in word clouds for each class and the limited effectiveness of word frequency analysis in capturing sarcasm, we decided to train our models using embeddings and other numerical representations. Embeddings provide a more nuanced representation of language by capturing semantic and contextual information. By leveraging embeddings, we aim to capture the subtle linguistic cues and contextual dependencies that are essential for sarcasm detection, enabling our models to better understand and distinguish sarcastic expressions within the dataset.

Training a SVM model

To investigate the impact of punctuation on our dataset and to establish a starting point for our analysis, we trained a simple **baseline model** using a Support Vector Machine (SVM) classifier with a Radial Basis Function kernel on TF-IDF representation of our data. Baseline models are often simpler and more interpretable, making them suitable for our initial experimentation. We created two versions of our dataset: one with punctuation cleaned and another without any punctuation cleaning. The goal was to compare the performance of the SVM classifier on these two versions and examine whether the inclusion or exclusion of punctuation had any effect on the model's accuracy or predictive capabilities. Obviously, our intention in using the baseline model was not to achieve state-of-the-art performance, but rather to explore the impact of punctuation before delving into more advanced models like BERT.

After conducting the experiment, we found that there was no significant difference in the performance of our simple baseline Support Vector Machine (SVM) classifier when trained on two versions of the dataset: one with punctuation cleaned and the other without any punctuation cleaning. The absence of a noticeable impact led us to the decision to discard the punctuation in our subsequent analyses. Despite the common belief that punctuation might carry sarcasm cues, our results suggest that for our specific task, the inclusion or exclusion of punctuation did not significantly affect the classifier's ability to detect sarcasm. As a result, we proceeded with the version of the dataset that had punctuation removed, focusing on other linguistic and contextual features to enhance our models.

Metric	Punctuation cleaned data	Punctuation data
Accuracy	0.78	0.78
F1 score	0.80	0.80
Precision	0.80	0.80
Recall	0.80	0.80

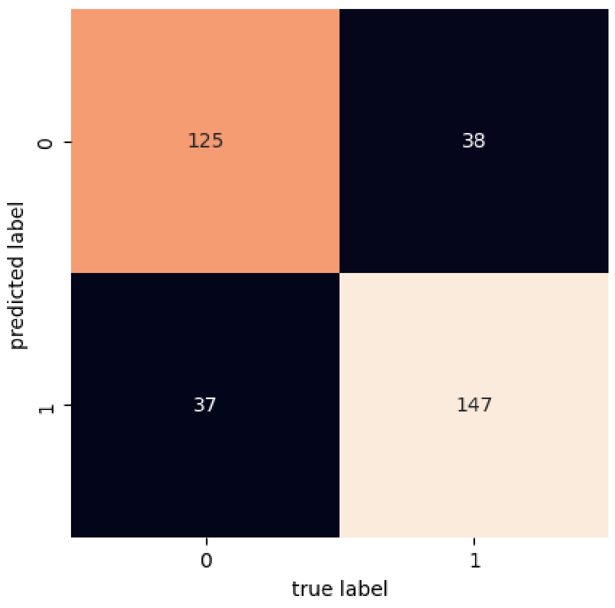


Figure 3: Confusion matrix for punctuation cleaned data

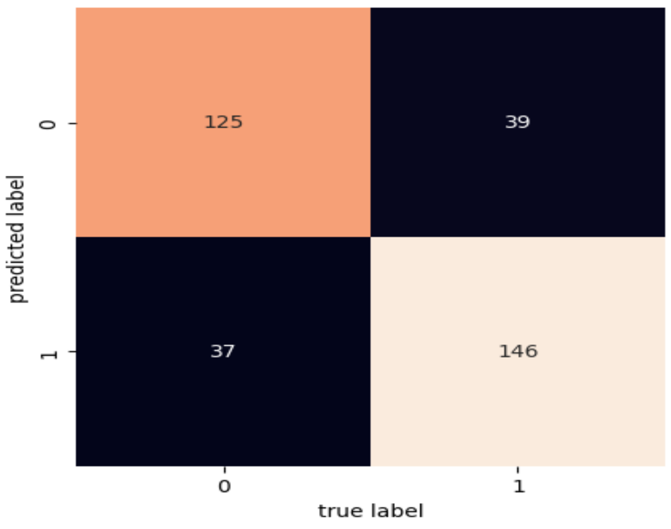


Figure 4: Confusion matrix for punctuation data

In our analysis, an important assumption we make is that the errors in our confusion matrix are balanced across different classes. This means that the misclassifications or errors made by the model are distributed fairly and do not favor any particular class more than others. Hence, we ensure that our data cleaning and analysis remains unbiased, allowing us to evaluate the overall performance of the model accurately, aiming to provide a reliable and objective assessment of the model's effectiveness in sarcasm detection.

Data Representation

Sentence Embeddings

GREEK-BERT for contextual embeddings:

GREEK-BERT was used for extracting the text embeddings. We decided to get the embeddings from the raw data of the dataset, in order to get insights for the effectiveness of the model in extracting suitable embeddings.

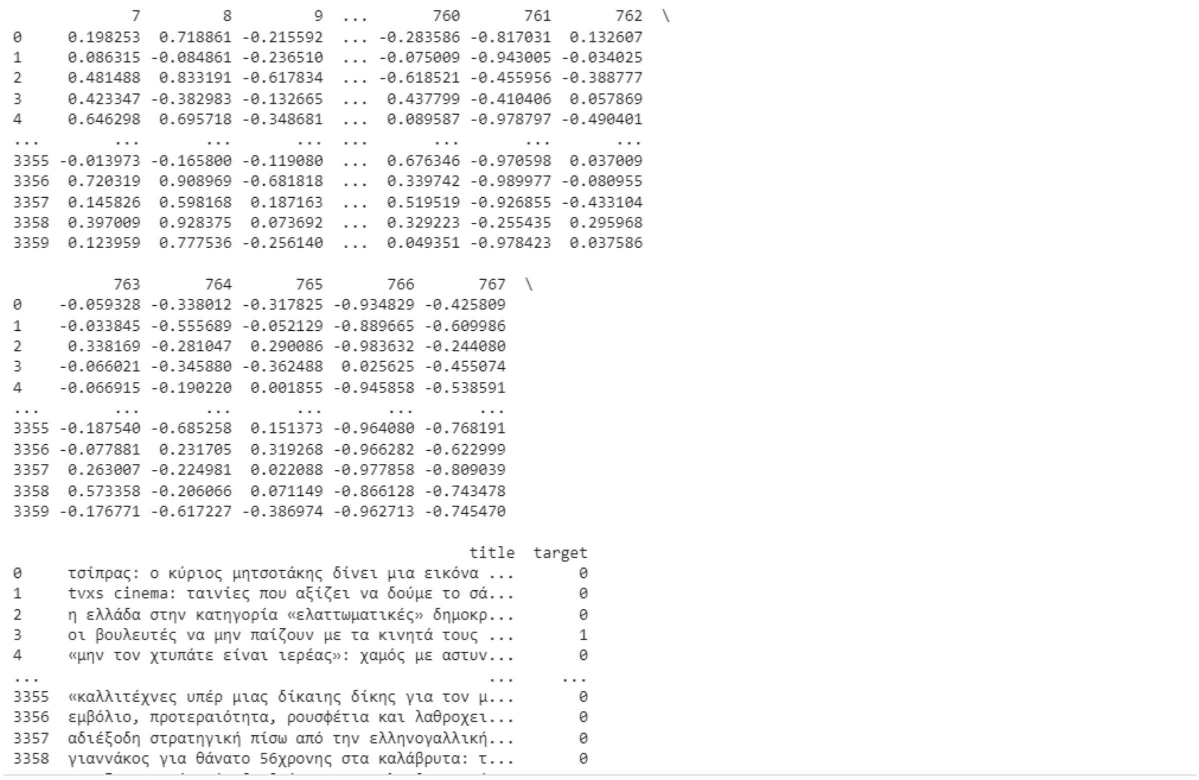
Generally, Transformer-based language models, such as BERT(Bidirectional Encoder Representations from Transformers) and its variants, achieve peak performance on NLP tasks, since they are pretrained on large amounts of unlabeled data, have a bidirectional approach (consideration of the entire context of a word by using both the left and right context), use self-attention mechanisms, are further-tuned after the training process and therefore produce high quality context-aware representations of sub-word tokens and entire sentences.

BERT undergoes pre-training using two auxiliary self-supervised tasks: (a) Masked Language Modeling (MLM), also known as Denoising Language Modeling, in which the model predicts hidden or masked tokens based on the contextual information provided by surrounding tokens, and (b) Next Sentence Prediction (NSP), where the model uses the [CLS] token representation to determine if sentence S2 immediately follows sentence S1 in the given corpus. Initially, four English models were introduced. Among them, BERT-BASE-UNCASED and BERT-LARGE-UNCASED convert all text to lowercase, while BERT-BASE-CASED and BERT-LARGE-CASED retain the original character casing.

For the GREEK-BERT specifically, the architecture of BERT-BASE-UNCASED was chosen, since the larger BERT-LARGE-UNCASED was computationally heavy for both pre-training and fine-tuning. Similar to the English Bert-base-uncased model, it has 12 layers, 768 units in its hidden layers, 12 parallel attention heads and 110 million learnable parameters. It was pre-trained on 29 GB of text from the following corpora: (a) the Greek part of Wikipedia, (b) the Greek part of the European Parliament Proceedings Parallel Corpus (Europarl) and (c) the Greek part of OSCAR (Open Super-large Crawled ALMAnaCH coRpus). Accents and other diacritics were removed along with the conversion of the text in lowercase, for the best possible normalization. Furthermore, a vocabulary of 35k BPEs was extracted with the SentencePiece library using the same corpora. For the pre-training of the model in the MLM and NSP tasks, the official code, provided by Google was used (1 million pre-training steps and the Adam optimizer were implemented with an initial learning rate of 1e-4 on a single Google Cloud TPU v3-8.7 Pre-training took approx. 5 days.)

Implementation:

The main tasks for extracting the embeddings, after processing the data by concatenating the two datasets (sarcastic and non sarcastic headlines) and shuffling them, are presented below. Initially, the necessary libraries were imported. In our case, the AutoTokenzizer and TFAutoModel classes were imported from the Transformers library for loading the pre-trained Tokenizer and Model of Bert. These two components are initialized using the relative functions and specifically use the *"nlpaueb/bert-base-greek-uncased-v1"* model, as a variant of the BERT model specifically trained for Greek text. After this, a sentence embedding function is defined, that takes a list of sentences as input. Inside the function, the sentences are encoded using the tokenizer with options for padding, truncation, and converting them into TensorFlow tensors. Then, the encoded sentences are passed through the model to obtain the sentence embeddings. The function returns the pooled output, which represents a fixed-size representation of each sentence. After extracting the titles and target labels (1= sarcastic, 0= non sarcastic) from our shuffled_df (It consists of both the datasets, sarcastic/non sarcastic, shuffled), the function is called and the “titles” list is passed as an argument, resulting in generating the sentence embeddings for each headline. Afterwards, a dataframe was created with the embeddings, their relative titles and target labels and saved for later use in our models’ training. The dataframe is displayed below:



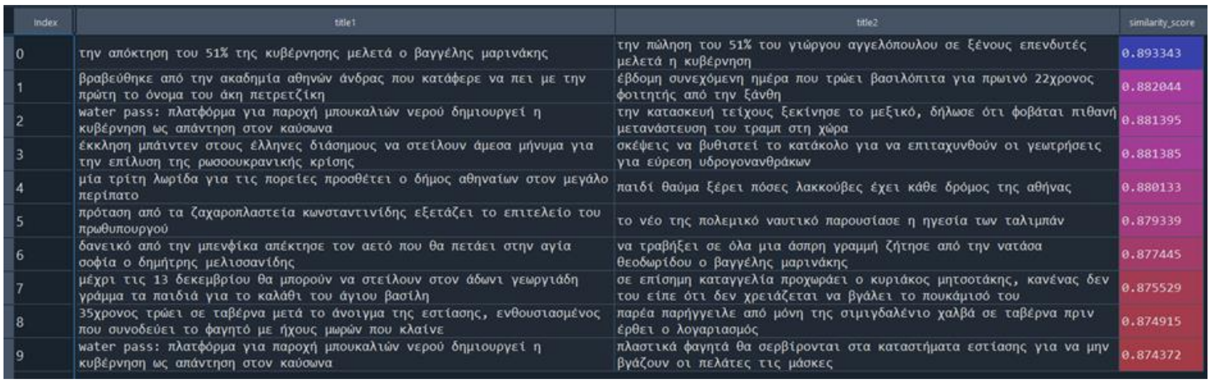
```
            7         8         9  ...       760       761       762  \
0     0.198253  0.718861 -0.215592  ... -0.283586 -0.817031  0.132607
1     0.086315 -0.084861 -0.236510  ... -0.075009 -0.943005 -0.034025
2     0.481488  0.833191 -0.617834  ... -0.618521 -0.455956 -0.388777
3     0.423347 -0.382983 -0.132665  ...  0.437799 -0.410406  0.057869
4     0.646298  0.695718 -0.348681  ...  0.089587 -0.978797 -0.490401
...        ...       ...       ...  ...       ...       ...       ...
3355 -0.013973 -0.165800 -0.119080  ...  0.676346 -0.970598  0.037009
3356  0.720319  0.908969 -0.681818  ...  0.339742 -0.989977 -0.080955
3357  0.145826  0.598168  0.187163  ...  0.519519 -0.926855 -0.433104
3358  0.397009  0.928375  0.073692  ...  0.329223 -0.255435  0.295968
3359  0.123959  0.777536 -0.256140  ...  0.049351 -0.978423  0.037586

           763       764       765       766       767  \
0    -0.059328 -0.338012 -0.317825 -0.934829 -0.425809
1    -0.033845 -0.555689 -0.052129 -0.889665 -0.609986
2     0.338169 -0.281047  0.290086 -0.983632 -0.244080
3    -0.066021 -0.345880 -0.362488  0.025625 -0.455074
4    -0.066915 -0.190220  0.001855 -0.945858 -0.538591
...        ...       ...       ...       ...       ...
3355 -0.187540 -0.685258  0.151373 -0.964080 -0.768191
3356 -0.077881  0.231705  0.319268 -0.966282 -0.622999
3357  0.263007 -0.224981  0.022088 -0.977858 -0.809039
3358  0.573358 -0.206066  0.071149 -0.866128 -0.743478
3359 -0.176771 -0.617227 -0.386974 -0.962713 -0.745470

                                                  title  target
0     τσίπρας: ο κύριος μητσοτάκης δίνει μια εικόνα ...       0
1     tvxs cinema: ταινίες που αξίζει να δούμε το σά...       0
2     η ελλάδα στην κατηγορία «ελαττωματικές» δημοκρ...       0
3     οι βουλευτές να μην παίζουν με τα κινητά τους ...       1
4     «μην τον χτυπάτε είναι ιερέας»: χαμός με αστυν...       0
...                                                 ...     ...
3355  «καλλιτέχνες υπέρ μιας δίκαιης δίκης για τον μ...       0
3356  εμβόλιο, προτεραιότητα, ρουσφέτια και λαθροχει...       0
3357  αδιέξοδη στρατηγική πίσω από την ελληνογαλλική...       0
3358  γιαννάκος για θάνατο 56χρονης στα καλάβρυτα: τ...       0
```

Data Exploration / Qualitative analysis

The embeddings extracted by using GREEK – BERT helped us gain some insight for our data. The two methods chosen for analyzing the results that the model provided is cosine similarity and clusters from KMeans.

Firstly, cosine similarity is a metric used to measure the similarity between two vectors in a high-dimensional space. In the context of sentence embeddings, cosine similarity is often employed to quantify the semantic similarity or relatedness between pairs of sentences. It is computed by taking the cosine of the angle between the vectors (the sentence embeddings are represented as vectors in a high-dimensional space and each vector corresponds to a specific feature of the sentence). It ranges from -1 to 1, where -1 indicates completely dissimilar vectors,1 points out identical vectors, and values in between represent varying degrees of similarity.

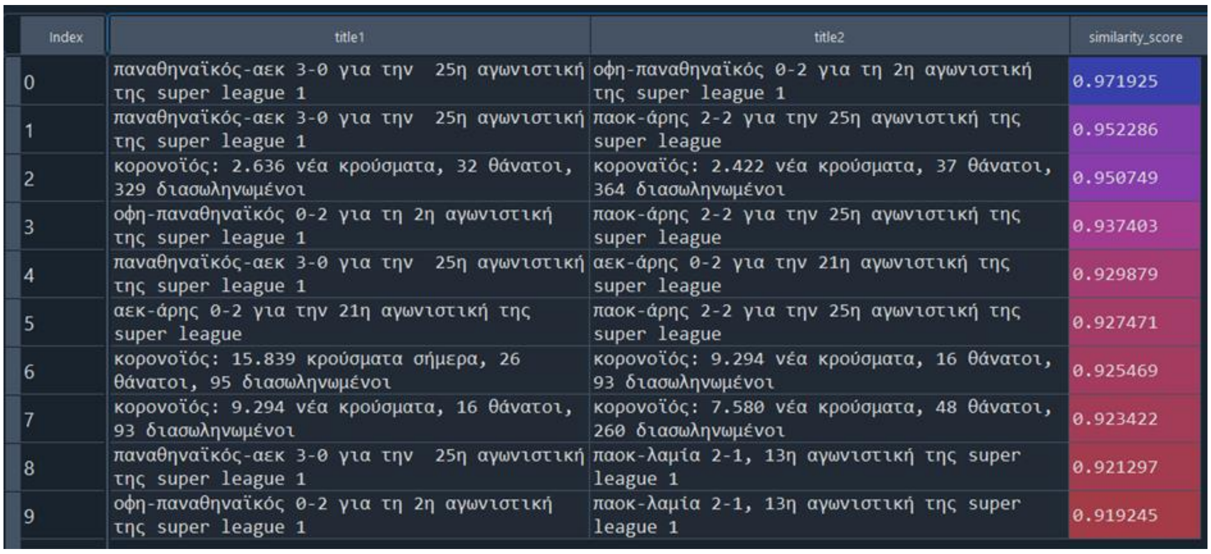
It was chosen as an appropriate metric in our case, since by using it, we can assess how similar or related two sentences can be based on their embeddings. With the help of a BERT model, we expect to capture the similarity in meaning, even if the sentences are worded differently or have different syntactic structures. As a result, we will be able to quantify and evaluate the similarity between the embeddings that GREEK-BERT gave us. The results are presented below:



Index	title1	title2	similarity_score
0	την απόκτηση του 51% της κυβέρνησης μελετά ο βαγγέλης μαρινάκης	την πώληση του 51% του γιώργου αγγελόπουλου σε ξένους επενδυτές μελετά η κυβέρνηση	0.893343
1	βραβεύθηκε από την ακαδημία αθηνών άνδρας που κατάφερε να πει με την πρώτη το όνομα του άκη πετρετζίκη	έβδομη συνεχόμενη ημέρα που τρώει βασιλόπιτα για πρωινό 22χρονος φοιτητής από την ξάνθη	0.882044
2	water pass: πλατφόρμα για παροχή μπουκαλιών νερού δημιουργεί η κυβέρνηση ως απάντηση στον καύσωνα	την κατασκευή τείχους ξεκίνησε το μεξικό, δήλωσε ότι φοβάται πιθανή μετανάστευση του τραμπ στη χώρα	0.881395
3	έκκληση μπάιντεν στους έλληνες διάσημους να στείλουν άμεσα μήνυμα για την επίλυση της ρωσοουκρανικής κρίσης	σκέψεις να βυθιστεί το κατάκολο για να επιταχυνθούν οι γεωτρήσεις για εύρεση υδρογονανθράκων	0.881385
4	μία τρίτη λωρίδα για τις πορείες προσθέτει ο δήμος αθηναίων στον μεγάλο περίπατο	παιδί θαύμα ξέρει πόσες λακκούβες έχει κάθε δρόμος της αθήνας	0.880133
5	πρόταση από τα ζαχαροπλαστεία κωνσταντινίδης εξετάζει το επιτελείο του πρωθυπουργού	το νέο της πολεμικό ναυτικό παρουσίασε η ηγεσία των ταλιμπάν	0.879339
6	δανεικό από την μπενφίκα απέκτησε τον αετό που θα πετάει στην αγία σοφία ο δημήτρης μελισσανίδης	να τραβήξει σε όλα μια άσπρη γραμμή ζήτησε από την νατάσα θεοδωρίδου ο βαγγέλης μαρινάκης	0.877445
7	μέχρι τις 13 δεκεμβρίου θα μπορούν να στείλουν στον άδωνι γεωργιάδη γράμμα τα παιδιά για το καλάθι του άγιου βασίλη	σε επίσημη καταγγελία προχωράει ο κυριάκος μητσοτάκης, κανένας δεν του είπε ότι δεν χρειάζεται να βγάλει το πουκάμισό του	0.875529
8	35χρονος τρώει σε ταβέρνα μετά το άνοιγμα της εστίασης, ενθουσιασμένος που συνοδεύει το φαγητό με ήχους μωρών που κλαίνε	παρέα παρήγγειλε από μόνη της σιμιγδαλένιο χαλβά σε ταβέρνα πριν έρθει ο λογαριασμός	0.874915
9	water pass: πλατφόρμα για παροχή μπουκαλιών νερού δημιουργεί η κυβέρνηση ως απάντηση στον καύσωνα	πλαστικά φαγητά θα σερβίρονται στα καταστήματα εστίασης για να μην βγάζουν οι πελάτες τις μάσκες	0.874372

Figure 5: Cosine similarity between sarcastic headlines

Inspecting the results that we got, we can say that it generally seems that GREEK-BERT has done a great job generating the sentence embeddings, since we can capture some actual semantic similarities between the headlines. For example, in pairs 0, 2, 3, 5 and 10, the idea of government's decisions (and more generally politics) in a sarcastic way, is detected. Pairs that are food related (1, 8) even though it's not totally visible, e.g. Άκης Πετρετζίκης, are successfully grouped into pairs with the same similarity. Furthermore, we can observe a successful rate of similarity detection between the single pairs. The highest similarity that was presented between the single pairs is 0.89. The two sentences indicate decisions in the government's field, with different individuals involved and include the exact same percentage as well.

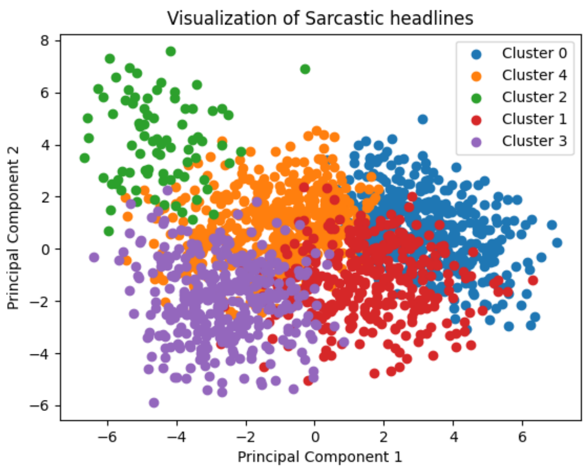


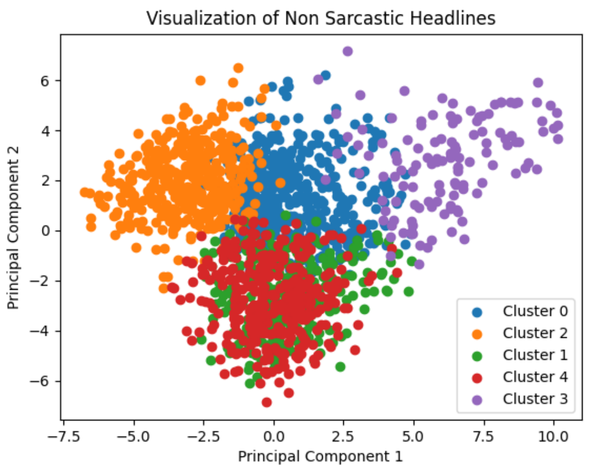
Index	title1	title2	similarity_score
0	παναθηναϊκός-αεκ 3-0 για την 25η αγωνιστική της super league 1	οφη-παναθηναϊκός 0-2 για τη 2η αγωνιστική της super league 1	0.971925
1	παναθηναϊκός-αεκ 3-0 για την 25η αγωνιστική της super league 1	παοκ-άρης 2-2 για την 25η αγωνιστική της super league	0.952286
2	κορονοϊός: 2.636 νέα κρούσματα, 32 θάνατοι, 329 διασωληνωμένοι	κοροναϊός: 2.422 νέα κρούσματα, 37 θάνατοι, 364 διασωληνωμένοι	0.950749
3	οφη-παναθηναϊκός 0-2 για τη 2η αγωνιστική της super league 1	παοκ-άρης 2-2 για την 25η αγωνιστική της super league	0.937403
4	παναθηναϊκός-αεκ 3-0 για την 25η αγωνιστική της super league 1	αεκ-άρης 0-2 για την 21η αγωνιστική της super league	0.929879
5	αεκ-άρης 0-2 για την 21η αγωνιστική της super league	παοκ-άρης 2-2 για την 25η αγωνιστική της super league	0.927471
6	κορονοϊός: 15.839 κρούσματα σήμερα, 26 θάνατοι, 95 διασωληνωμένοι	κορονοϊός: 9.294 νέα κρούσματα, 16 θάνατοι, 93 διασωληνωμένοι	0.925469
7	κορονοϊός: 9.294 νέα κρούσματα, 16 θάνατοι, 93 διασωληνωμένοι	κορονοϊός: 7.580 νέα κρούσματα, 48 θάνατοι, 260 διασωληνωμένοι	0.923422
8	παναθηναϊκός-αεκ 3-0 για την 25η αγωνιστική της super league 1	παοκ-λαμία 2-1, 13η αγωνιστική της super league 1	0.921297
9	οφη-παναθηναϊκός 0-2 για τη 2η αγωνιστική της super league 1	παοκ-λαμία 2-1, 13η αγωνιστική της super league 1	0.919245

Figure 6: Cosine similarity between non-sarcastic headlines

Regarding the cosine similarity between our non-sarcastic headlines, we could say that the results were interesting. Firstly, we can see that the similarity scores are much higher than the sarcastic headlines, with the highest pair being at 0.97. However, looking more closely at the individual pairs, we can understand why they achieve such high scores. The headlines are extremely identical, reaching a point where the only thing that changes is the numbers. In all 10 most similar sentences, the subjects that are mentioned are only 2 (sports and covid). This suggests that for this particular dataset we have limited subject variety. The non sarcastic headlines seem to be focused only on particular topics and lack diversity. This constitutes a limitation of our dataset, which should be taken into consideration for anyone who plans to use it for any NLP projects.

Our second way of inspecting our data is the implementation and visualization of clusters in our dataset with KMeans. Clustering with K-means is a popular unsupervised machine learning algorithm used for grouping similar data points together based on their feature similarity. The goal of K-means clustering is to partition the data into K distinct clusters, where K is a user-defined parameter representing the number of clusters desired. In our case, we chose 5 clusters for each label of the headlines (sarcastic, non-sarcastic) to check out the grouping between the headlines and visualize them. Firstly, the K initial cluster centroids are randomly selected as the representatives or centers of the clusters. Each data point is assigned to the nearest centroid based on Euclidean distance. Then, the centroids of the clusters are recalculated by computing the mean (center) of all the data points assigned to each cluster. This process is repeated until convergence is met. Convergence occurs when the assignments of data points to clusters no longer change or when a maximum number of iterations is reached.

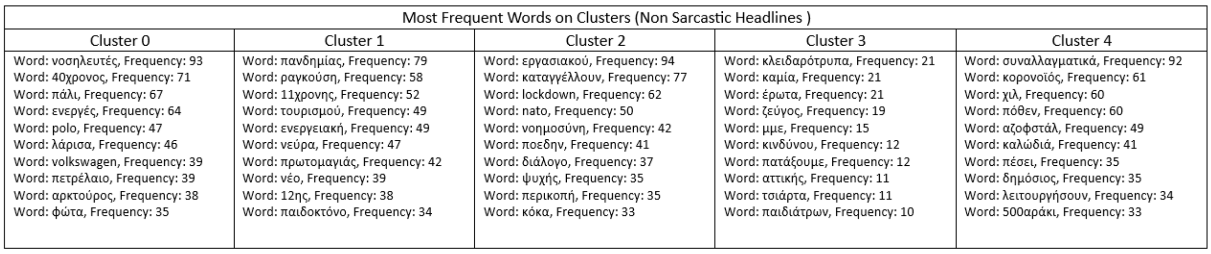




The separation of the 5 clusters in the sarcastic headlines suggests that there are distinct topics or themes in the dataset. As a result, each cluster represents a different aspect or category of sarcasm. The relatively separate clusters in the sarcastic sentences imply a higher level of diversity in terms of sarcastic expression. This shows that there may be different ways in which sarcasm is passed, with distinct patterns or linguistic cues associated inside the different clusters. On the other hand, the non-sarcastic headlines are not distinctly separated. The plot shows that the groups are more overlayed between them, especially cluster 1 and 4. The overlap between clusters in the non-sarcastic sentences implies that there are more similarities between different non-sarcastic headlines, as it is mentioned above, in the cosine similarity. This indicates that the non-sarcastic headlines probably share common characteristics, such as language patterns, reporting styles, or popular topics.

Indicatively, the most frequent words for each cluster are presented below, for checking out the direction of the main themes inside the clusters:

Most Frequent Words on Clusters (Sarcastic Headlines)				
Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Word: βλέποντας, Frequency: 176 Word: επισημαίνουν, Frequency: 144 Word: πιτσαρίας, Frequency: 135 Word: ευρωπαίους, Frequency: 122 Word: καθαρίσει, Frequency: 109 Word: αναγνώρισε, Frequency: 106 Word: επιθυμία, Frequency: 103 Word: παγώνη, Frequency: 94 Word: γεγονός, Frequency: 94 Word: θάλασσες, Frequency: 93	Word: λογαριασμούς, Frequency: 121 Word: κάπνισε, Frequency: 120 Word: κορέα, Frequency: 83 Word: έξαλλος, Frequency: 82 Word: εκτέθηκε, Frequency: 76 Word: πρωινής, Frequency: 74 Word: εταιρεία, Frequency: 71 Word: κατέθεσε, Frequency: 71 Word: συγκεντρώνει, Frequency: 64 Word: πρωτοεμφανιζόμενος, Frequency: 58	Word: σου, Frequency: 21 Word: περισσότερα, Frequency: 21 Word: peanut, Frequency: 15 Word: λέξεις, Frequency: 15 Word: βιβλίο, Frequency: 12 Word: φωτογραφία, Frequency: 12 Word: εταιρεία, Frequency: 11 Word: παραγράφους, Frequency: 11 Word: παρατάει, Frequency: 11 Word: γλώσσα, Frequency: 11	Word: μαρμαρωμένου, Frequency: 115 Word: ντόναλντ, Frequency: 108 Word: σας, Frequency: 86 Word: story, Frequency: 67 Word: κλικ, Frequency: 65 Word: λέει, Frequency: 63 Word: πουλέν, Frequency: 56 Word: 23ωρο, Frequency: 56 Word: καταλάβαιναν, Frequency: 49 Word: διακόψει, Frequency: 46	Word: μερίδα, Frequency: 179 Word: καλεσμένο, Frequency: 175 Word: εργαζόμενος, Frequency: 156 Word: συνεργασία, Frequency: 121 Word: ελένης, Frequency: 114 Word: sms, Frequency: 108 Word: άλμπουμ, Frequency: 103 Word: παραγγελία, Frequency: 97 Word: 24χρονος, Frequency: 91 Word: πρωταγωνιστών, Frequency: 87



Most Frequent Words on Clusters (Non Sarcastic Headlines)				
Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Word: νοσηλευτές, Frequency: 93	Word: πανδημίας, Frequency: 79	Word: εργασιακού, Frequency: 94	Word: κλειδαρότρυπα, Frequency: 21	Word: συναλλαγματικά, Frequency: 92
Word: 40χρονος, Frequency: 71	Word: ραγκούση, Frequency: 58	Word: καταγγέλλουν, Frequency: 77	Word: καμία, Frequency: 21	Word: κορονοϊός, Frequency: 61
Word: πάλι, Frequency: 67	Word: 11χρονης, Frequency: 52	Word: lockdown, Frequency: 62	Word: έρωτα, Frequency: 21	Word: χιλ, Frequency: 60
Word: ενεργές, Frequency: 64	Word: τουρισμού, Frequency: 49	Word: nato, Frequency: 50	Word: ζεύγος, Frequency: 19	Word: πόθεν, Frequency: 60
Word: polo, Frequency: 47	Word: ενεργειακή, Frequency: 49	Word: νοημοσύνη, Frequency: 42	Word: μμε, Frequency: 15	Word: αζοφστάλ, Frequency: 49
Word: λάρισα, Frequency: 46	Word: νεύρα, Frequency: 47	Word: ποεδην, Frequency: 41	Word: κινδύνου, Frequency: 12	Word: καλώδιά, Frequency: 41
Word: volkswagen, Frequency: 39	Word: πρωτομαγιάς, Frequency: 42	Word: διάλογο, Frequency: 37	Word: πατάξουμε, Frequency: 12	Word: πέσει, Frequency: 35
Word: πετρέλαιο, Frequency: 39	Word: νέο, Frequency: 39	Word: ψυχής, Frequency: 35	Word: αττικής, Frequency: 11	Word: δημόσιος, Frequency: 35
Word: αρκτούρος, Frequency: 38	Word: 12ης, Frequency: 38	Word: περικοπή, Frequency: 35	Word: τσιάρτα, Frequency: 11	Word: λειτουργήσουν, Frequency: 34
Word: φώτα, Frequency: 35	Word: παιδοκτόνο, Frequency: 34	Word: κόκα, Frequency: 33	Word: παιδιάτρων, Frequency: 10	Word: 500αράκι, Frequency: 33

Data Augmentation

In our effort to extract better results, we tried data augmentation, reaching 3.000 headlines for each category, with a total augmented dataset of 6.000 headlines. By augmenting the dataset with additional samples, we could possibly achieve a better balance between the classes. This could be beneficial for the training models, as it gives them a wider range of data as input, so they have a better chance of generalizing more appropriately to unseen data. As a result, we would expect our model to get a bit more robust, since it will learn the features across multiple instances (in this way, the augmented embeddings can create different data scenarios). So, we decided to try running our models with the augmented data, in order to check if we would notice any significant differences in their performance. The results of the model with the augmented data, along with the balanced ones are presented in a table below.

Implementation

Methods

- **Neural Networks**

Deep Learning Frameworks

For the task implementation Tensorflow, Keras and Pytorch were used.
For the neural network experiments, the following architectures were used:
MLP (Multi-Layer Perceptron), BiLSTM (Bidirectional Long Short Term Memory) including (or not) an Attention layer, and BiGRU (Bidirectional Gated Recurrent Unit)

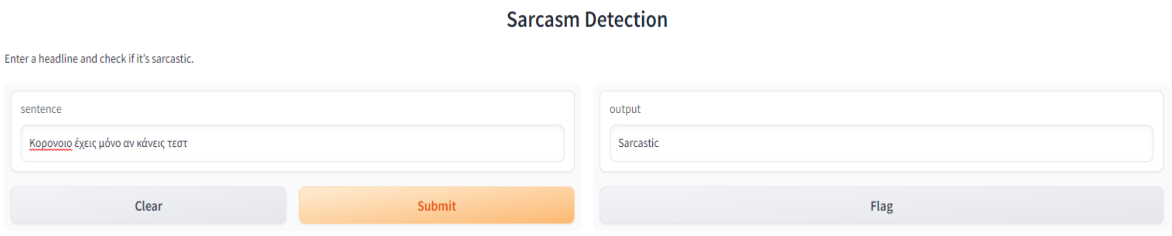
MLP, which is a simple version of an Artificial Neural Network, gives us a first intuition about the training of our data and the classification performance.

LSTMs are effective at capturing long-term temporal dependencies without suffering from the optimisation hurdles that plague simple learning models. Especially, the Bidirectional LSTM, enables training by capturing connections from both forward and backward context of the sentences.
We examine the addition of an **Attention mechanism** in the biLSTM architecture, in order to enable the model to focus on different parts of the input based on their importance or relevance. We created a simple attention mechanism, which calculates attention weights based on a dense layer applied to the input tensor and then applies those weights to the input tensor, resulting in a weighted representation.
Finally, we explore the gating mechanism in a Recurrent Neural Network. The **GRU** is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate. Since, it is believed that in some tasks, GRU performs better than LSTM we wanted to explore, if that could be the case for our task too.
All the above architectures were trained with the BERT embeddings, which have large dimensions (768), and allow the model to gain common knowledge and improve the model performance.

Layers	Model #1: MLP	Model #2: Bi-LSTM	Model #3: Bi-GRU	Model #4: Bi-LSTM+Attention
1	Dense (64 units, ReLU, (input))	Bidirectional LSTM (32 units, 0.2 dropout)	Bidirectional GRU (128 units, 0.2 dropout)	Bidirectional LSTM (32 units,0.2 dropout)
2	Dropout (0.3)	Dense (256 units, ReLU)	Bidirectional GRU (64 units)	Attention()
3	Dense (32 units, ReLU)	Dense (128 units, ReLU)	Dropout (0.3)	Dense (256 units, ReLU)
4	Dropout (0.2)	Dropout (0.3)	Dense (32 units, ReLU)	Dense (128 units, ReLU)
5	Dense (16 units, ReLU)	Dense (10 units, ReLU)	Dropout (0.3)	Dropout (0.3)
6	Dense (2 units, softmax)	Dense (1 unit, Sigmoid)	Dense (1 unit, Sigmoid)	Dense (10 units, ReLU)
7				Dense (1 unit, Sigmoid)

- **Greek BERT Fine-tuning**
  Apart from using the pretrained Greek BERT to generate sentence embeddings for our data, we also fine-tuned Greek BERT on our dataset for our sequence classification task. In particular, the process involved converting the data into InputExamples, processing them into InputFeatures, creating a TensorFlow dataset, compiling the model, training it on the dataset and this process gave us the highest results. The BERT model is compiled using the Adam optimizer with a learning rate of 3e-5, an epsilon value of 1e-08, and gradient clipping with a norm of 1.0.
  The loss function is set to SparseCategoricalCrossentropy, which is suitable for multi-class classification problems.
  The idea for adding these specific hyperparameters was taken from a Sentiment Analysis task on Skroutz Customer Reviews using the Greek BERT.

By replicating the code of this notebook found in Kaggle[1] in our data, we unexpectedly reached extreme high results in our train set, so we decided to keep it as our reference code, based on which we created an app using gradio interface.


Sarcasm Detection

Enter a headline and check if it's sarcastic.

sentence: Κορονοιο έχεις μόνο αν κάνεις τεστ

output: Sarcastic

Results

Classification Reports

Undersampled

Model	**Accuracy**	**Precision** (macro averaged)	**Recall** (macro averaged)	**F1** (macro averaged)
SVM(Radial Basis Function kernel)	0.78	0.80	0.80	0.80
MLP	0.79	0.75	0.85	0.80
Bi-LSTM	0.77	0.79	0.77	0.77
Bi-GRU	0.80	0.81	0.81	0.81
Bi-LSTM + Attention	0.80	0.80	0.80	0.80
Greek BERT fine-tuning	**0.93**	**0.94**	**0.92**	**0.93**

[1] https://www.kaggle.com/code/nikosfragkis/skroutz-sentiment-analysis-with-bert-greek

Augmented

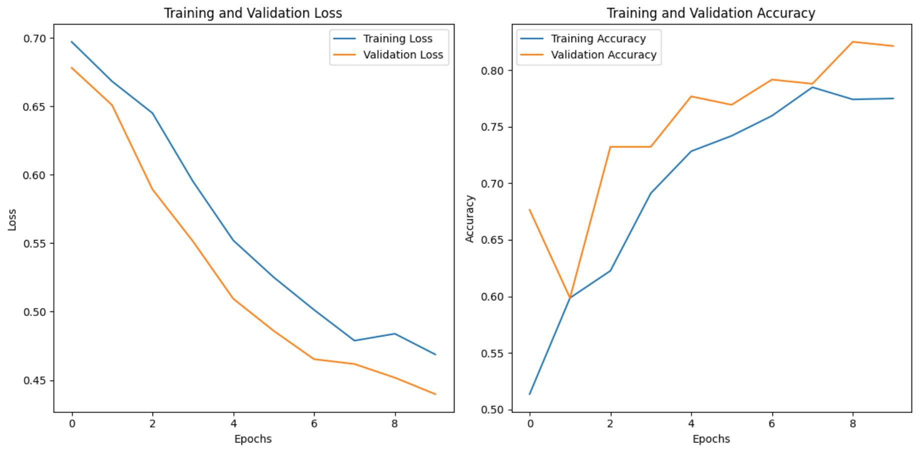
Model	**Accuracy**	**Precision** (macro averaged)	**Recall** (macro averaged)	**F1** (macro averaged)
MLP	0.82	0.84	0.78	0.81
Bi-LSTM	0.85	0.85	0.85	0.85
Bi-GRU	0.83	0.84	0.84	0.84
Bi-LSTM + Attention	0.85	0.86	0.85	0.85
Greek BERT fine-tuning	**0.99**	**0.98**	**0.99**	**0.99**

Comparison

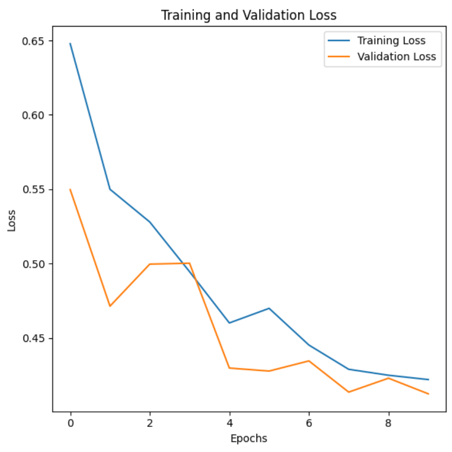
Learning Curves:

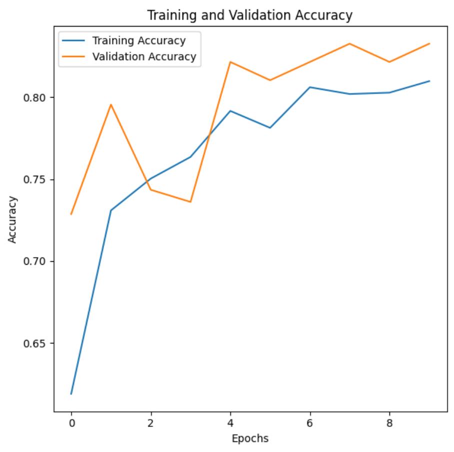
Undersampled:

MLP

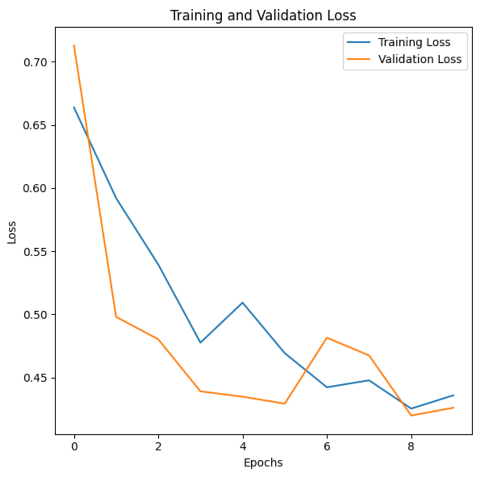


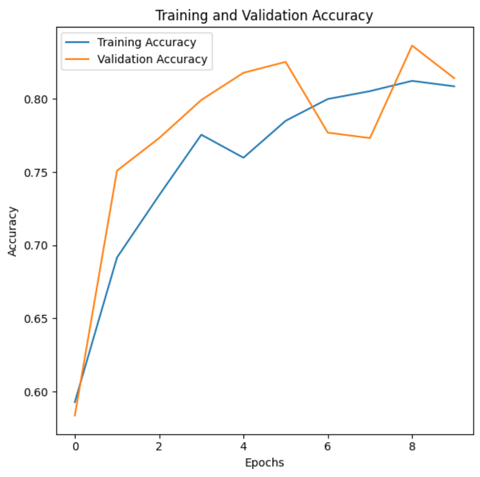
bi-LSTM



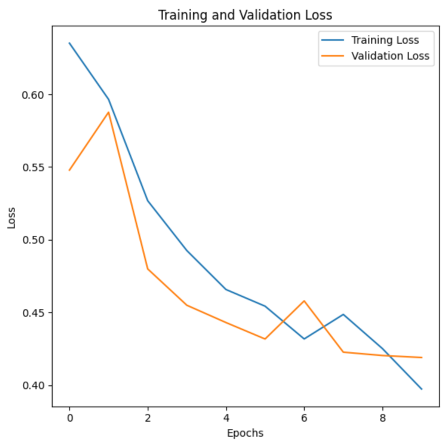


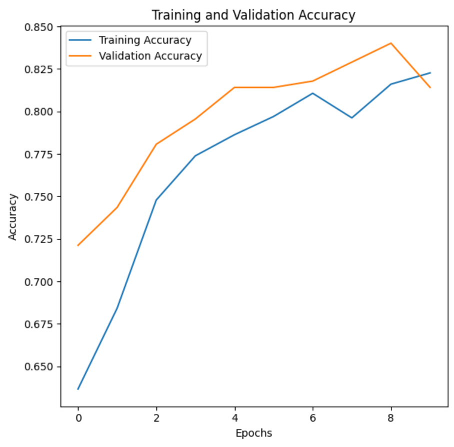
biGRU





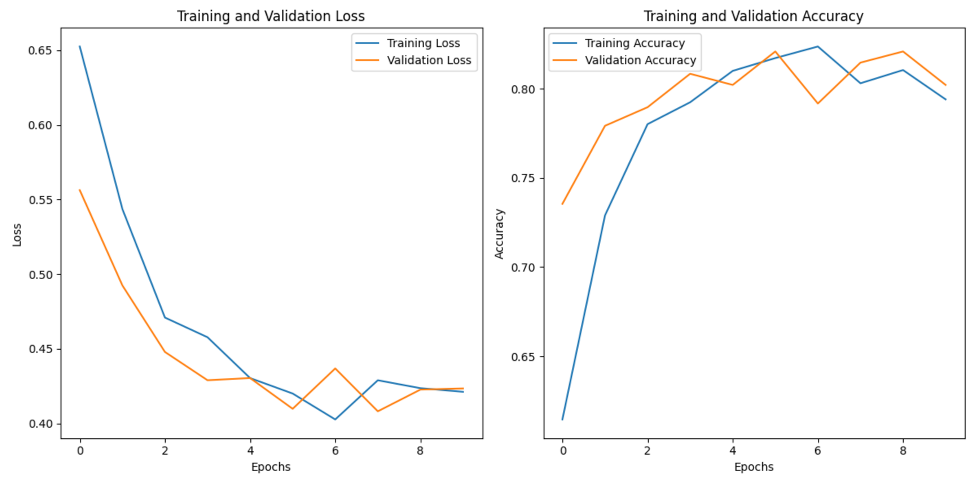
biLSTM+Attention



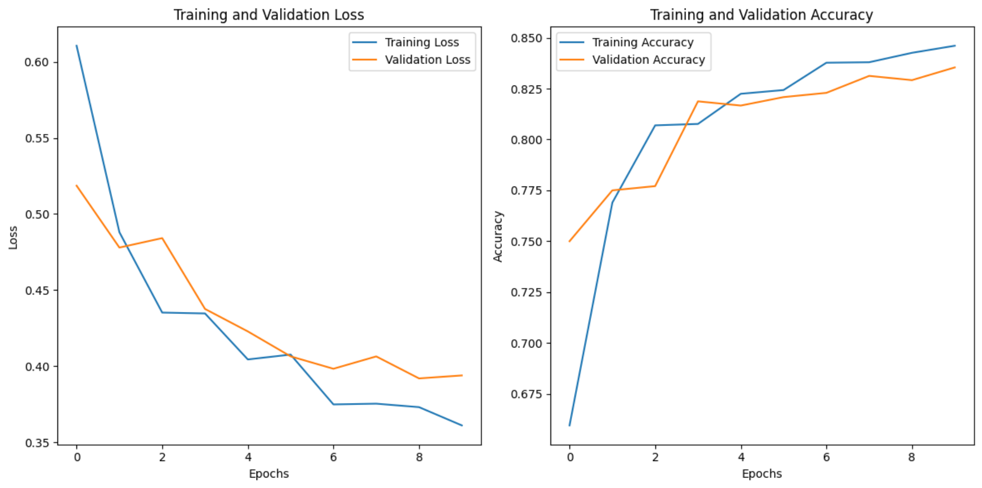


Augmented:

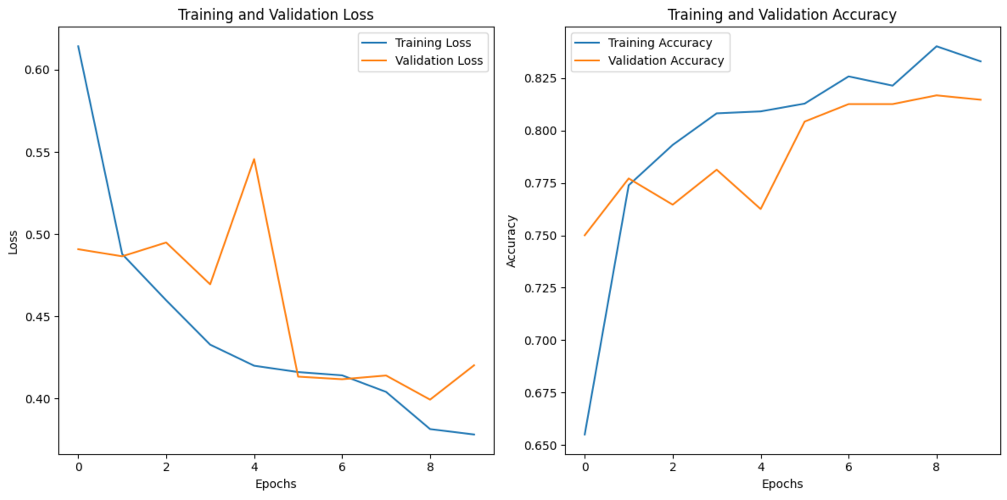
MLP



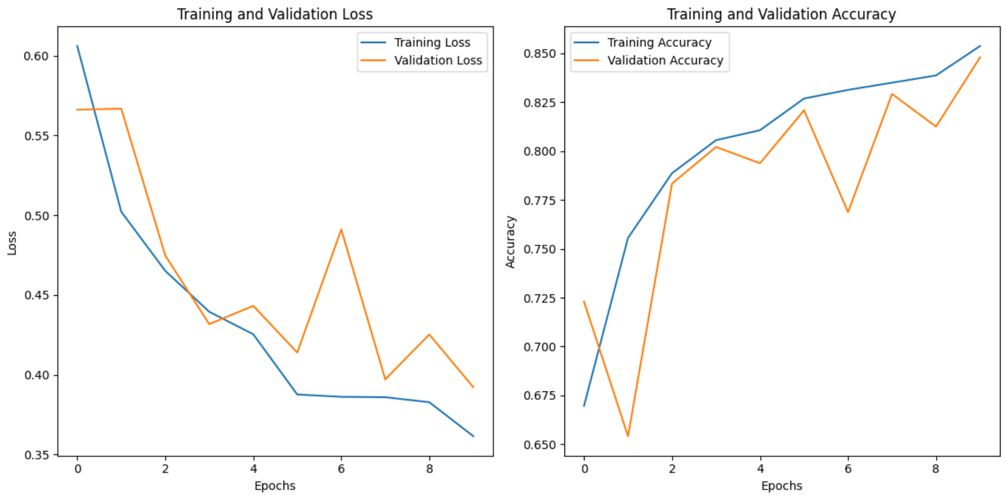
biLSTM



biGRU



biLSTM+Attention



Conclusions

- The use of Greek BERT proved to be very beneficial for our specific task, since it allowed us to extract sentence embeddings and consequently capture their contextual meaning, which is significant for NLP projects such as classifying sarcastic and non-sarcastic texts. Besides helping us gain insight on the raw data that were collected (exploration of clusters), it also generated appropriate embeddings for training the models. This is proven by the performance of the models, which was very sufficient. Overall, our models achieved an accuracy between 0.78 (SVM) and 0.99 (Greek BERT fine-tuning).
- Data augmentation indeed improved the average model's performance, reaching a 5% increase in all the examined models.
- The Attention Mechanism seemed to not be contributing to the model's performance. This may be the case, because our custom attention layer focuses on the importance of individual elements within the input sequence but does not explicitly model the relationships between different positions.
- No neural Network architecture indicated an outstanding performance, which leaves us only with the fine-tuned BERT producing exceptionally high results.
- The predictions of the fine-tuned BERT seemed to be almost identical with the true labels on the test set, whereas the predictions when using random sarcastic and not sarcastic sentences of Greek news websites in the app were less accurate. This proves our assumption that sarcasm is a linguistic phenomenon strongly correlated with the cultural landscape and current affairs describing a specific year or period of time. It can be concluded that for a more accurate performance on the task of sarcasm detection, algorithms need to be constantly fed with more recent data, at least concerning the news-headlines framework.

Future work

- Try more complex attention mechanisms, like self-attention, which focuses on capturing relationships within a sequence or multihead attention, which captures different types of information or patterns at different positions in the input sequence by attending to different parts of the sequence simultaneously.
- Apply more feature engineering techniques in our data.Specifically, in our case,it would be beneficial to conduct a deeper analysis of the non-sarcastic headlines, in order to identify patterns and similarities beyond their surface-level content. Techniques such as topic modeling could help with understanding the data and gain insight on how they could be diversified with different variations.
- Expand the generated dataset with new data like sarcastic tweets, in order to enhance the detector’s range of applications.
- Apply NLP methods for further Analysis of the Dataset, like (NER, Event Extraction and Topic Modelling)

References

- Azwar, A. S., & Suharjito. (2020). Sarcasm Detection Using Multi-Channel Attention Based BLSTM on News Headline. https://doi.org/10.21203/rs.3.rs-63423/v1
- Misra, Rishabh and Prahal Arora. "Sarcasm Detection using News Headlines Dataset." AI Open (2023)

Links to notebooks/articles

- https://towardsdatascience.com/sarcasm-detection-with-nlp-cbff1723f69a
- Sarcasm Detection in News Headlines - YouTube
- https://www.youtube.com/watch?v=P1RMRa2sl48&ab_channel=TinyMLStudioCainvas
- Sarcasm Analysis | Kaggle
- News Articles in Greek | Kaggle
- Topic Modeling in Greek News using ML techniques | Kaggle https://towardsdatascience.com/the-5-most-useful-techniques-to-handle-imbalanced-datasets-6cdba096d55a
- https://www.darpa.mil/news-events/2021-05-06
- https://www.kaggle.com/code/nikosfragkis/skroutz-sentiment-analysis-with-bert-greek
- https://github.com/codebasics/deep-learning-keras-tf-tutorial/blob/master/47_BERT_text_classification/BERT_email_classification-handle-imbalance.ipynb
- https://towardsdatascience.com/sarcasm-detection-with-nlp-cbff1723f69a
- https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%201%20-%20Lesson%203.ipynb
- https://towardsdatascience.com/creating-a-multilayer-perceptron-mlp-classifier-model-to-identify-handwritten-digits-9bac1b16fe10
- https://github.com/royjafari/DataAnalyticsForFun/blob/main/MLP%20Classification/MLP%20Classify%20-%20E.ipynb
- https://github.com/nlpaueb/greek-bert
- https://arxiv.org/pdf/2008.12014.pdf