Web Content Classification Analysis

Nissopoulou Theopisti Xeni

SID: 3308200019

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

JANUARY 2023

THESSALONIKI – GREECE

INTERNATIONAL
HELLENIC
UNIVERSITY

Web Content Classification Analysis

Nissopoulou Theopisti Xeni

SID: 3308200019

Supervisor: Academic Associate Dimitrios Karapiperis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

JANUARY 2023

THESSALONIKI – GREECE

Abstract

This dissertation was written as a part of the MSc in Data Science at the International Hellenic University.

Web analytics is a way for companies to learn more about the people who visit their websites. This information may include things like how people use the pages on a site, and how they respond to different types of content. These types of analytics can help to determine future decisions regarding the content and marketing which may help on how the company is perceived from the customers, as well as it can even improve their status and the profit. In order to gain valuable insights from a series of site visits and other related interaction data, it is essential to have accurate data. In the current work, Web Content Text Classification is to be done over the extracted content of a company's portal by categorizing those into 19 brands. Natural Language Processing techniques and Machine Learning algorithms have been applied and described. After evaluating the results of the models, it was concluded that BERT, which is a powerful deep learning model, and in particular Bert Base Uncased, performed the best, making accurate predictions and having good performance overall.

Nissopoulou Theopisti Xeni

January, 2023

Abbreviations

ANN Artificial Neural Networks

API Application Programming Interface

BERT Bidirectional Encoder Representations from Transformers

BOW Bag-of- Words

BPE Byte-Pair-Encoding

CALA ClAssifying Links Automatically

CBOW Continuous Bag of Words

CNN Convolutional Neural Networks

DANN Deep Artificial Neural Networks

DL Deep Learning

DNN Deep Neural Networks

DTM Document-Term Matrix

FFN Feed-Forward Network

GRU Gated Recurrent Unit

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

KBDI Knowledge Based Deep Inception

LSTM Long Short-Term Memory

ML Machine Learning

MLM Masked Language Modeling

MSE Mean Squared Error

NN Neural Network

NLP Natural Language Process

OOV Out-Of-Vocabulary

OWPCM Objectionable Web Page Classification Method

POA Parliamentary Optimization Algorithm

POS Part-of-speech

RL Reinforcement Learning

RNN Recurrent Neural Networks

SGD Stochastic Gradient Descent

SOP Sentence Ordering Prediction

TFI-DF Term Frequency - Inverse Document Frequency

URL Uniform Resource Locator

WPCA Web page classification algorithm

WWW World Wide Web

Contents

1 Introduction

According to Sethunya et al. [1], Natural Language Process (NLP) is explained across literature as the research area in which, applications are used to test how well computers can read, comprehend, and respond to natural language content in order to solve a variety of issues. Text classification is one of the many tasks under the wider variety of applications in the NLP area. The technique of classifying text into categories or classes, based on content, with the use of machine learning algorithms and methods is called text classification. Classification in general, is the supervised learning technique where a so-called classifier is trained over an already labeled set of objectives, trying to achieve the highest of accuracy results once it is called to classify new objects in one of the categories/classes that it has been trained over. The purpose of this exercise is to perform text classification in the content of web pages. That means that, an additional technique is required in order to obtain the content from those pages. This technique is called web scraping. Bo [2], defines web scraping (or else called web extraction/harvesting) as the technique to extract data from the World Wide Web (WWW) in order to process it in any way.

The focus of this analysis will be over a company's portal. The structure of the company is made into business units and brands (products). Marketeers and analysts of each branch worldwide, use customers' information over the portal's activity by analyzing them over dashboards, to understand behaviors, get insights about them in order to segment customers and better target them, get learnings about preferences and best practices for future campaigns. One of the issues that has been identified is that, many web pages are misclassified with regards to their business unit / brand. This issue makes data unreliable in terms of insights and actions to be taken. The process of manually identifying the correct class of every possible page over the portal and change it requires limitless working hours and also involves a significant margin of human error. Another alternative is to re-create all the already existing pages with the exact same content under a new URL and with a new tag for the class. This process also involves risks for the company, as marketing material that has already been shared with the customers will no longer be accessible and this will drive to a loss of interaction and engagement with the audience. This is the reason why a machine learning approach will be examined. By this approach, the company can

continue creating content and new, correctly tagged web pages, undisturbed while focusing on training the relevant employees on how this should be done correctly. The miss-tagged data will be handled and corrected over the consumption interaction layer of data, so that the correct info is used across all the end-user applications. The purpose of this exercise is to acquire the best possible results over the correct classes. Currently, over 80% of the portal content is untagged (miss-tagged) and we aim to an improvement over 20%. Some risks that will be faced are that many of the provided pages that have been created on company's portal over the last 2 years might not be active anymore and also, some of those might not be accessible without logging in with account credentials.

Over the first Chapter there is a literature review of similar work that has been done over the text classification as well as web content classification in particular. Most of the milestones and the evolution of the fields are mentioned along with a reference over their performance over time.

The second chapter focuses on the theoretical background of the core of the thesis, which is the Machine Learning area. The three main categories of Machine Learning are described with a deeper focus on the one that fits the purpose of the current work.

The definition of Deep Learning lays over the third chapter, where the categories of the Neural Networks and the Encoders/Decoders are described.

Transformers and in particular their architecture and some of the models that have been developed on top of them and are also used for this work are presented on the fourth chapter.

The fifth chapter consists of Natural Language Processing definitions and techniques regarding text preprocessing, vectorization, and augmentation.

Over the last chapter, the case study of this thesis is described, including the statement of the challenge, the approach that is followed in order to achieve the final goal, the description of the dataset that has been used and how the data have been handled. The question about which models have been chosen and why are also addressed along with the final results. The findings of this work and recommendations for future studies are presented in the final section of this chapter.

2 Related work

Over this chapter the general concept of text classification at first and then the web content classification concept in particular will be reviewed over the existing literature. Different approaches of web content classification techniques based on what each one achieves and where does each one focuses will be also analyzed. Lastly, approaches on how to extract content from the web pages will be inspected. As a last steps, the classifiers that have been used in such relevant cases will be analyzed.

2.1 Text classification

Classification in general, is a supervised machine learning which can be described as the process of an algorithm (classifier) that is fed and trained with a variety of data, where each one of those is already classified, in order to be capable to predict the class of previously unseen data.

Sebastiani [3], defined text categorization as the process of "assigning a Boolean value to each pair $(d_j, c_i) \in D \times C$, where D is a domain of documents and $C = \{c_1, \ldots, c_{|C|}\}$ is a set of predefined categories". It is also stated that the objective is to "approximate a target function that describes how documents ought to classify".

2.2 Web content classification

Web content classification is very similar to the text classification approach, even though it is more complicated. Its complexity is depending on the erratic nature of the construction of the web pages and the way that data are derived in order to be analyzed. The number of variables, the HyperText Markup Language (HTML) tags that are included, the hyperlinks, the various formats, the noise, the advertisement banners, etc. are some of the elements included over the data on the web pages which demand various pre-processing steps before text can be fed into a classifier.

Once content from web is obtained and pre-processed, data are usually represented in multi-dimensional vectors with various types. The dimensionality will be extremely high if every feature is taken and turned into a vector, which will be very consuming in terms of time and space and complexity. That's why, steps for feature selection and dimensionality reductions are also part of the web content classification tasks.

In 2013, Patel et al. [4], proposed Objectionable Web Page Classification Method (OWPCM), a filtering system for objectionable web documents, as one supervised approach which classifies documents into 2 categories, objectionable and non-objectionable. This approach included the training of a Neural Network (NN) algorithm, entropy term weighting indexing algorithm, and Principal component analysis (PCA) feature reduction algorithm. This technique was evaluated over 4 datasets and achieved an accuracy >92% in every case.

In 2014, Hernández et al. [5], introduced an unsupervised approach (so called ClAssifying Links Automatically - CALA) for web page classification by clustering the web pages from a site so that each cluster represents a set of a unique class. The article states that none of the existing proposals were appropriate for integrating companies web information, as they did not meet requirements such as lightweight crawling and the ability to avoid training on pre-classified pages (as most approaches rely on supervised techniques). CALA creates Uniform Resource Locator (URL) patterns that correspond to different classes of pages on a website, allowing new pages to be classified by fitting their URLs to those patterns. Another paper that was published in 2014 by Asheghi et al. [6], compared a variety of content-based features for web page and found that lexical features performed better than all other features. On the same paper, a semi-supervised mini-cut algorithm approach was tested, which employed the pages that are underneath of the target web pages in hierarchy, as unlabeled data. The findings of this approach, which takes advantage of the graph structure of the pages, showed that some classes were more benefited than others.

In 2016, Kathirvalavakumar et al.[7], provided a new weighting approach that recognizes the significant and distinctive terms in a web page to lower the quantity of significant data that can aid the classifier in making a more accurate classification decision by reducing the dimensionality.

In 2017, Kiziloluk et al. [8], used the Parliamentary Optimization Algorithm (POA), which is one of the latest social-based metaheuristic algorithms, for Web page classification. The POA results were compared to other algorithms and were found to be higher. This study was the first to suggest the POA Web page classification using HTML tags as features.

In 2020, Aydos et al. [9], introduces a method which combines multiple NN in order to classify the web pages. The idea of this approach is to focus on the web page classification

problem using Google Image Search results as descriptive images. In this approach, each element is represented by multiple descriptive images and after the training process, each one of them is assigned a class based on the calculation of the descriptive image results.

In 2021, Gupta et al. [10], proposed an ensemble Knowledge Based Deep Inception (KBDI) for classifying web pages. This approach involved learning bidirectional contextual representation using pre-trained BERT incorporating Knowledge Graph embeddings and fine-tuning the objective task by applying Deep Inception network using parallel multi-scale semantics. The proposed ensemble tests the effectiveness of combining specialized domain knowledge embeddings with the pre-trained BERT model. The suggested BERT fused KBDI model performs better than benchmark baselines and other traditional techniques tested on web page categorization datasets, according to experimental interpretation.

In 2022, Kurt et al. [11], used Deep Learning (DL) approaches in order to classify web pages through their text content by constructing binary and multi-class classification models. In this study, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) DL techniques were used. Instead of n-gram techniques, word embedding was employed for feature extraction and hyper-parameter optimization was performed. The results of this study achieved an average f1 score of 90% with CNN, 89% with LSTM and 89% with GRU over the binary classification and 78% with CNN, 76% with LSTM and 77% with GRU over the multi-class classification.

In 2022, Yu [12] published a paper exploring a web page classification algorithm (WPCA) based on deep learning (DL), as the author believed that previous research on WPCA based on DL was not thorough. The article first utilizes a keyword weight calculation approach to first minimize the influence of a few high-frequency words on the weight calculation and lower the value of low-frequency word weights, improving the calculation accuracy of the WPCA. The classification technique, in addition, defines the category of all texts based on the similarity between the text to be classed and all of the class templates as well as certain classification criteria. Finally, the optimization method automatically modifies the size of the learning rate in order to enhance the DL learning rate. According to the experimental findings, WPCA based on DL is quicker, more effective, and uses less system compared to traditional algorithms.

Even though many specialized approaches have been made and developed on web classification techniques as detailed above, an evolution has been made, including, but not

limited to Natural Language Processing when Transformers where introduced. In 2017, Vaswani et all. [13] published a paper with the title ‘Attention is all you need’ where they claim that by then, the leading sequence transduction models relied on encoder-decoder setup and large or complex recurrent or convolutional neural networks. The top-performing models also connected the encoder to the decoder though an attention mechanism. They suggested a newly introduced, uncomplicated network design called the Transformer that relies purely on attention processes and discards both recurrence and convolutions. Their publication purely disrupted the Machine Learning community as since then, many applications, ideas and research are based on the foundation of Transformers on Computer Vision, Natural Language Processing and many more. Heavily dependent on Transformers, various of pre-trained models simplify many tasks by transfer learning. Based on the latter and given that the goal of this thesis is to accurately classify web pages which consist at the most of text, all the terms around Transformers and transfer learning will be analyzed and then attempted for the specific task over the next chapters.

3 Machine Learning

Over this chapter, the wider concept of Machine Learning will be introduced as it is the foundational component of all the concepts associated with the object of the thesis.

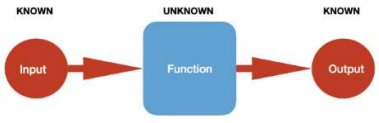
Machine Learning (ML) is defined as the usage and design of computer systems that can adapt and learn without being supplied explicit instructions, by analyzing historical data or data patterns in order to make predictions or assumptions using statistical models and algorithms. The above-mentioned definition can be interpreted as that, ML, entails constructing and training simple or complex models, using labeled or non-labeled datasets, in an effort to minimize error rates on the outcome/result of the model, when previously unseen data are fed into it.

The complexity and learning curve of the algorithm are two significant process metrics, but the quality of the input data is what matters most. Since a learning algorithm's objective depends on the input data, statistics and machine learning are closely connected fields. However, their primary distinction is that whereas ML focuses on predicting patterns, statistics seeks to discover links between variables and the significance of those associations.

The three most high-level and abstract subcategories of machine learning are: supervised, unsupervised, and reinforcement learning. Particularly, activities like Regression and Classification are under the purview of supervised learning, whereas Clustering and Policy Learning go under unsupervised learning and, finally, reinforcement learning respectively. The next sections will go into high-level detail on each of these concepts.

3.1 Supervised Learning

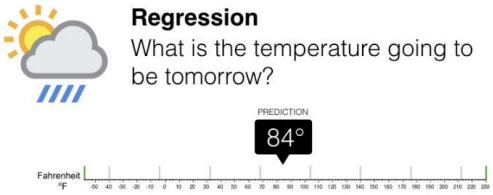
Supervised learning is a method of learning the input-output relationship of a system using a set of labeled training samples, where the output is known as the label of the input data. These input-output training samples are also referred to as labeled training data or supervised data.



Picture 1: Supervised Learning [1]

3.1.1 Regression

A Regression model is a way of predicting a specific outcome based on a group of input variables. Since the target value is a continuous value within a specific range, regression aims to predict the result of unseen records within that scale. There are various regression techniques available, and they can be categorized according to how many independent variables there are and how strongly the relationship between the independent variables is.



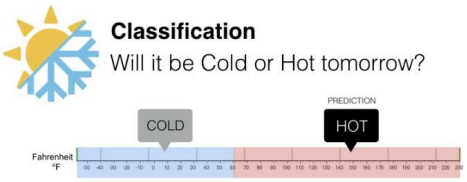
Picture 2: Regression example [2]

The performance of a regression model on how accurately it is able to predict previously unseen data, which, in other words can be phrased as how well does the model generalizes, is evaluated with metrics. One of the metrics that can be calculated in order to evaluate a regression model is Mean Squared Error (MSE) [14]. MSE calculation shows the squared difference between the actual and the predicted value thus, the lower the MSE of the model the better the model is performing. Another evaluation metric is R^2, also called coefficient of determination. R^2 evaluates the scatter of data points around the best fit line, thus, the higher the value, the better the model is performing.

3.1.2 Classification

When the objective fits into a group of discrete values known as labels, categories, or classes, classification is utilized. A classification model predicts the probability of an example being classified into a specific class as a continuous value. Classification tasks can either be binary (the input data are allocated between two classes, so the value obtained after classifying the data would be either 0 or 1, yes or no etc.,), multi-class (the input data are allocated across more than two classes) or multi-label (every data point of the input data would belong into more than one classes).

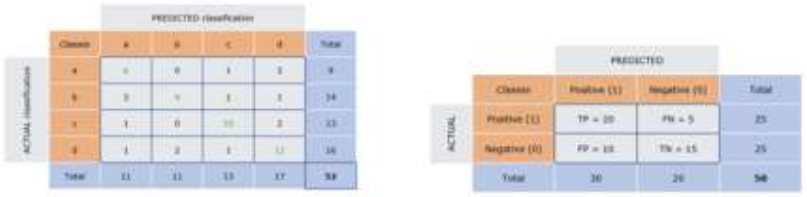
As described above, the main distinction between classification and regression approaches lays on the type of the outcome, and whether the problem somebody is trying to solve requires a continuous rather than a discrete prediction. On the other hand, we can approach the classification issue as a regression problem in a non-Euclidean space.



Picture 3: Classification example [1]

Many metrics that are used to evaluate the performance of a classification model, are based on the Confusion Matrix [15].

One of the most popular metrics in order to evaluate a classifier is Accuracy. Accuracy considers the sum of the True Positive and True Negative predictions as the numerator and the sum of all the entries of the Confusion Matrix as the denominator (the same applies for binary, multi-class, and multi-label classifiers. There are several evaluation metrics for every type of classification task such as Precision, Recall, F1 Score, Micro F1-Score, Macro F1-Score [16] etc.. The decision about which one should be monitored when the model is trained or fine-tuned so to rank the model and achieve the better possible results relies on the objective of the question that we are trying to answer.

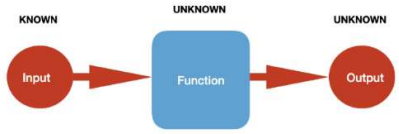


Picture 4: Confusion Matrices [3]

3.2 Unsupervised Learning

Machine learning algorithms are used in unsupervised learning to organize and evaluate unlabeled datasets. Without hu-man assistance, these algorithms find hidden patterns or

data clusters. It is helpful for exploratory data analysis since it may find similarities and contrasts in data as well as in various other real-world challenges.



Picture 5: Unsupervised Learning [1]

Unsupervised Learning can be also used as an ML tool. Data don't often come in perfect shape and form in order to be fed an algorithm and be transformed to give the desired outcome. When the challenge is the high dimensionality of the dataset, an unsupervised ML technique can be used in order to reduce it while at the same time keep as much of the data variation's details as possible.

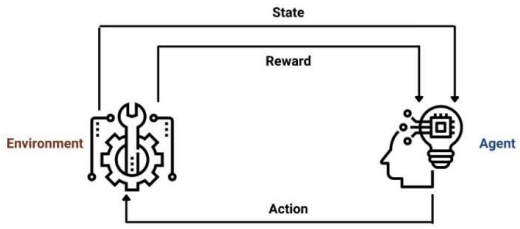
Two further well-liked techniques are self-supervised learning and semi-supervised learning. A technique known as semi-supervised learning combines a sizable amount of unlabeled data with a little amount of labeled data in order to draw insights from them. When there is a dearth of labeled data, there isn't enough domain expertise to label the data, or there isn't enough time to label the data, this method is frequently employed. On the other hand, self-supervised learning refers to situations when the model uses the natural structure of the data rather than labeled data to anticipate the output. This means that massive, unstructured datasets may be effectively mined for valuable information using self-supervised learning algorithms.

3.3 Reinforcement Learning

The machine learning technique known as reinforcement learning (RL) trains one or more agents in an environment via rewards. The agents can operate in the environment, and their performance determines whether they receive rewards or punishments. It's common practice to teach agents to seek long-term, maximum benefits from their surroundings by successfully performing tasks. The idea of policy learning is to instruct agents on the appropriate course of action via the use of punishment and rewards. The simulation's agents watch the environment as it is and behave accordingly. The agents proceed to the following state, but with the costs of their actions. After that, the action that has been

performed is evaluated and they compare the reward/punishment policy to find the best way to achieve the most profit.

In conclusion, RL differs from supervised and unsupervised machine learning approaches in the way that it operates. The agent's actions result in feedback from the environment. The labels used in supervised learning are not required as the agent is rewarded or penalized in accordance with their behavior.



Picture 6: Reinforcement Learning [1]

ML has become more advanced, encompassing a wide range of fields and domains. This has been powered by advancements in the field itself and in the hardware available. Based on the type of input data, there are different types of fields that have emerged, such as NLP for textual data, Computer Vision for visual data, and Speech Processing for phonetic data. Each field of data allows for the identification of tasks and problems which are currently being actively researched, as well as any potential solutions which may be available. Regardless the field and the data type, the main objective is to train a machine learning model that outputs a desired outcome. Training refers to the process of adjusting the model so that it produces the desired outcome as accurately as possible. In order to complete each task, the definition of the approach, outcome, and data will all be necessary. The type and architecture of the model will also be determined, and the method which can be used will also be determined.

Given the topic of this thesis, over the next chapter, techniques on the field of NLP will be described.

4 Deep Learning

Deep Learning is a form of ML that was developed due to the increasing need of better models that are able to accurately predict future events out of big amounts of data. Both operate just similarly. Although the machine learning models can learn to gradually improve, they will need human supervision to make sure they are getting the most out of their learning. DL models learn how to predict future outcomes and excel in doing so.

Predicting what will happen requires taking in input, processing it, and then producing a desired output. In DL, features are extracted from data using a variety of techniques, as opposed to Machine Learning where the required features are chosen, by people.

4.1 Training Deep Artificial Neural Networks

Deep Artificial Neural Networks (DANN) are computer programs that simulate the human brain in order to understand complex tasks better. The network is made up of multiple basic components, known as neurons, that are coupled to one another and organized in a certain topology [17]. A neuron in a hidden layer processes a number of input signals, and gives an output signal. The input and output signals are known as activations because they cause neurons to fire. A neuron's axon links to an additional neuron by a synapse.

There are several phases involved in training a (DNN). Synapses transmit data from the network's input layer to the hidden layer in DNN learning. Synapses are weighted. After being activated, the weighted sum plus the bias is sent as input to the neurons of the following hidden layer. Each synapse is given a new weight, which aids in computing the activation function and creating an output. The linked neurons in the hidden layer collaborate to produce a response, which is then transmitted back to the hidden layer and continued until the output layer is reached, which is the ultimate objective.

The network assigns random values to the weights when input data is first introduced to it. Since the model does not know anything about the data, it cannot determine its weights. To create the required output, the neural networks must modify the weight values. Gradient descent is a type of hill-climbing optimization that is used to update the weights. Gradient descent requires knowledge of the gradient of the loss function, or the vector holding the partial derivative of the loss function with respect to each of the parameters. Although

we need to backpropagate the layers to make up for the loss of the hidden layers, we can still make progress.

Neural networks come in a variety of forms and are employed for various tasks. Some popular deep neural networks are multi-layer perceptrons, convolutional, and recurrent neural networks. Some of the types that are important as concepts for the purposes of this thesis will be briefly mentioned below.

4.1.1 Feed-Forward Networks

A Feed-Forward Network (FFN) is a network that consists of more than one hidden layers and the neurons are solely linked to those in the layers following. These networks can be found is tasks like pattern recognition, computer vision, noise filtering, and more.

4.1.2 Convolutional Neural Networks

A convolutional layer and a fully connected layer make up convolutional neural networks (CNNs). The Convolutional layers are made up of convolutions that hold valuable information. Following every convolution, the most crucial traits are maintained. The output of the fully connected layers is imported as input of the convolutional layers, which will then output the outcome. CNN may be used for Image Recognition tasks, Video Analysis and other applications.

4.1.3 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are FFN that are rolled out over time and are made to accept a number of inputs, which denotes that each input of the sequence has some relation or influence with/on the neighbors. Basic FFN "remember" things as well, but only those they have already learnt. RNN learn similarly during training, but they additionally retain information from previous inputs while producing output (s). RNN require a lot of time to be trained and also deal with the issue of the vanishing gradient descent, where the gradient of prior time steps doesn't continue to grow, and therefore the network can't learn as well as it could. Applications that use machine translation, time series predictions, speech recognition, and music synthesis are based on RNNs. The Vanishing Gradient descent issue is solved by using a RNN (Long short-term memory (LSTM) and Gated recurrent units (GRUs)) together. LSTM networks have a memory that can be reused. They are able to recall prior knowledge that is crucial to the learning process. They have the ability to only keep information that is necessary for its purpose. Two

implementations of a LSTM network are speech recognition and writing recognition. An adaptation of LSTMs, GRUs feature 3 gates: an update gate, a reset gate, and a current memory gate.

4.1.4 Encoders and Decoders

A form of sequence-to-sequence model based on recursive neural networks (RNNs) is the encoder and decoder. At each time stamp, the encoder receives a word vector from the input sequence and the hidden state of the prior time step, updating the hidden state to store information about the input sequence. The last encoder unit's hidden state is subsequently transmitted to the decoder. Additional details over specific concepts that are required for the comprehension and the implementation of the subject of this thesis will be described over the next chapters.

5 Transformers

The majority of issues in the NLP field of activities, such text categorization and translation, are sequence related. RNNs have been utilized in several applications, however it has been shown that they have issues primarily because they cannot be trained in parallel, which extends the time required to handle a large dataset and makes implementation inefficient.

A new model was developed that can properly anticipate and address a variety of NLP issues in the paper 'Attention is all you need' [13]. The transformer is a deep learning model that utilizes a self-attention mechanism and is mainly used in the area of NLP. The machine depends entirely on its own attention to calculate its representations of its input and output. The transformer can process the whole sequence as input, and the data do not need to be labeled.

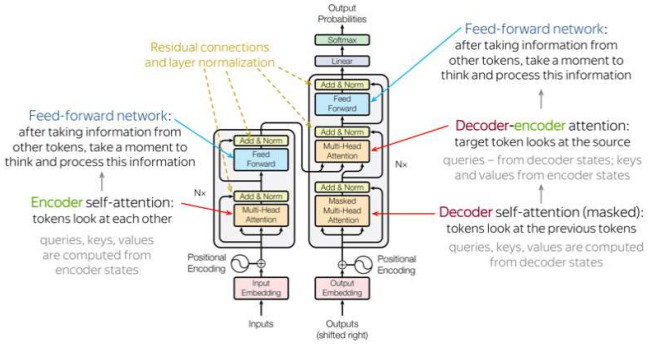
5.1 Architecture

According to Vaswani et al. [13], a transformer architecture consists of various encoders and decoders, plied on top of each other. Every encoder consists of a multi-head attention layer followed by a feed forward neural network layer. The decoder, on the other hand, is composed of a masked multi-head attention layer, a multi-head attention layer, a feed forward neural network layer, and a final layer.

The number of the encoders and the decoder units is a hyperparameter but the number of them is always equal to each other. Word embeddings are accepted as input and are processed by each encoder before being processed by the final encoder. The first decoder and the succeeding decoders receive the output of the final encoder as input. The extra masked multi-head attention layer aids in sharpening focus on crucial sequence elements.

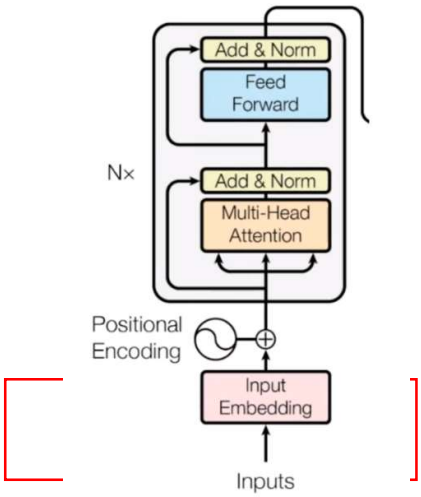
Every encoder and decoder includes a self-attention layer, which helps to clarify specific phrases in the sequence. The query, key, and value vectors are constructed in each encoder and updated during training. Self-attention is calculated for each word, and to calculate self-attention for the first word, all other word scores are also calculated in relation to the first word. Once all the terms in the sequence have been processed, the next words are processed using this method.

Utilizing the SoftMax activation function, each score is normalized. The normalized scores are then multiplied by the relevant value vectors and added together to create the final vector, which is the result of the feed-forward network layer's output from the self-attention layer. Each decoder and encoder has multiple self-attention layers, hence the term 'multi-head attention,' each of which computes an output. Each self-attention layer is carried out independently and concurrently. The feed-forward network receives a final concatenation of all the outputs.



Picture 7: Transformers: Model Architecture [4]

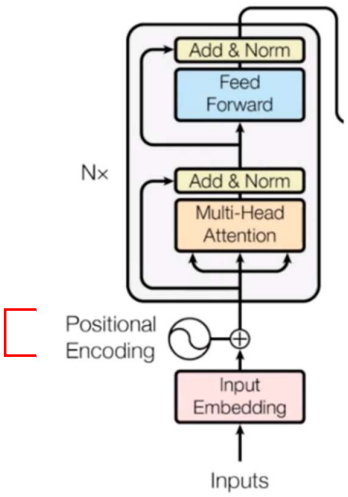
5.1.1 The Encoder block



Input Embedding

In order to transform words into embedding vectors the concept of embedding space is introduced. This can be perceived as a dictionary where words after being transformed into tokens of a fixed vocabulary, those of similar meaning are grouped together. Every word inside this embedding space is mapped and given a specific value based on what it means.

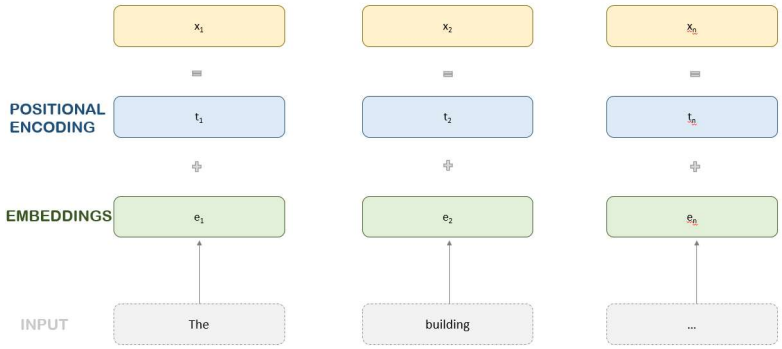
Picture 8: Input Embedding [4]



Positional Encoding

The sequence, as referenced before, is processed all at once. Words though, may take different meaning based on the sentence they are in. Each token embedding is given a positional encoding vector by the transformer so as to determine the position of the tokens in the sequence, resulting in a special embedding with positional information.

Picture 9: Positional Embedding [4]

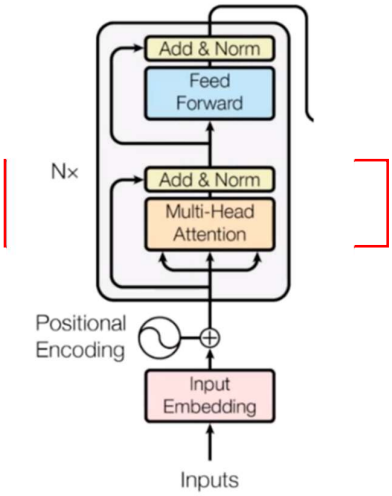


Picture 10: Process of Input data (*picture created by the author*)

These procedures prepare the vectors, which are presented as context, for consumption by the encoders. In order to produce a new vector per token with the same structure as the input sequence but enhanced with more complicated information, the encoder receives one vector for each token in the sequence. This procedure will be broken out step by step below.

Multi-Head Attention

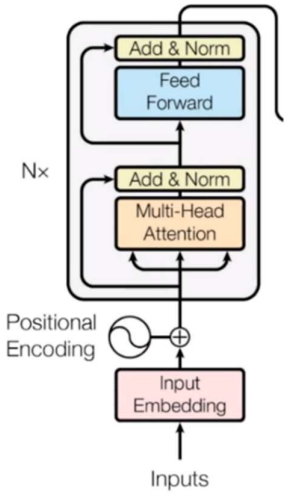
The Multi-Head Attention, which is the fundamental element of the transformers and is also known as self-attention, is the following process and the initial layer of the encoder. No matter how far apart, this layer may identify related tokens in the same sequence. As an attention vector, it focuses on the word's importance in relation to the other words in the phrase. An attention vector is generated for every word and it captures the contextual relationship among the words in a sentence. In order to address the case where for every word, its value is weighted more on itself, a multiple attention vectors per word are determined and the weighted average is computed at the end. This process is called multi-head attention block as multiple attention vectors are used.



Picture 11: Multi-Head Attention , Add & Norm [4]

Add & Norm

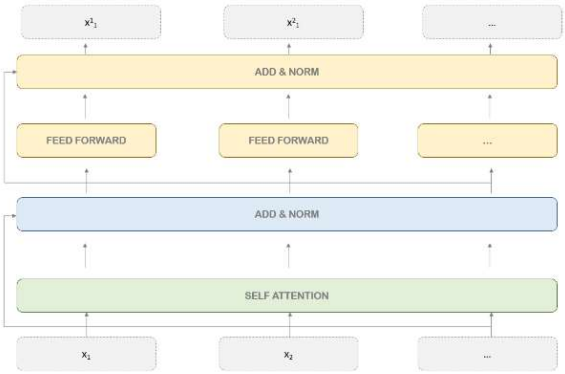
Add & Normalize are in fact two separate steps. The add step is a residual connection. The idea was introduced by He et al. in 2005 [18] with the ResNet model. It is one of the solutions for vanishing gradient problem. The normalization step is about layer normalization which was introduced by Ba et al., in 2016 [19], it is a way of normalization.



Picture 12: Feed-Forward Network [4]

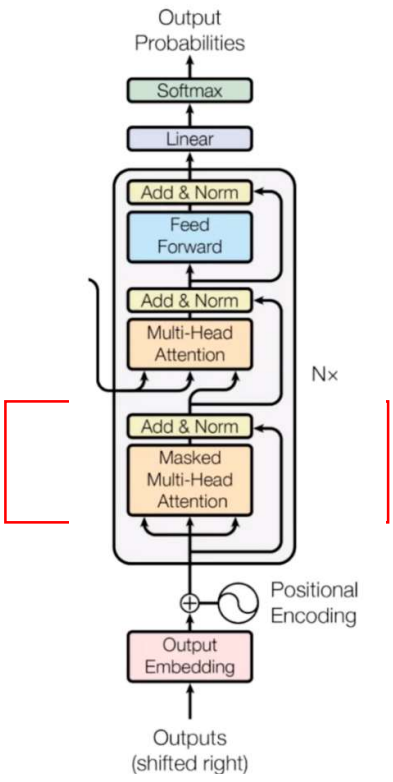
Feed-Forward Network

The Feed Forward network as well as an additional Add & Normalize phase are included in the encoder's final stage. Each attention vector token is processed by a feed-forward neural network in parallel. One attention vector at a time is accepted by the feed-forward network. The greatest aspect is that each of these attention vectors is independent of the others, unlike the RNN example. Therefore, we may use parallelization in this situation, and this is one of the differences that Transformers introduced in contradiction to the traditional RNN. At the end, an additional Add & Normalize step is applied with the input and output of the feed-forward step.



Picture 12: Output of the Encoder block (*picture created by the author*)

5.1.2 The Decoder block



Picture 13: Output Embedding and Positional Encoding[4]

Output Embedding and Positional Encoding

The decoder block is actually an encoder only with an additional encoder-decoder attention layer. If the model is training a translator for Greek to English, a Greek sentence has to be given along with an its equivalent in English. The Greek sentence will pass through the encoder and the English sentence will pas through the decoder.

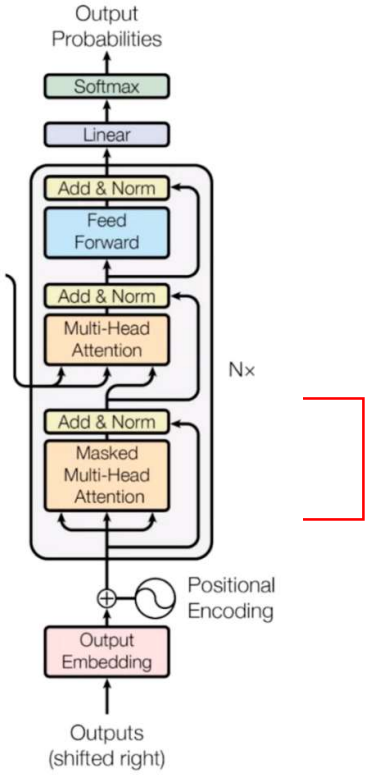
The first phase is the same as the one that is explained on the encoder block, and it consists of the positional encoding and embedding layers that convert the words into the appropriate vectors.

Masked Multi-Head Attention

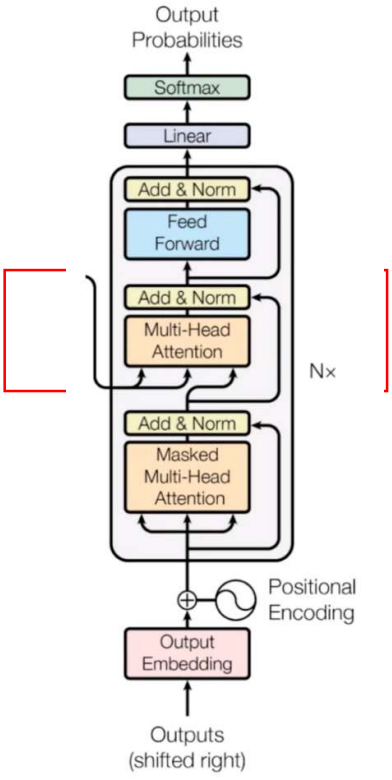
The Masked Multi-Head Attention Layer is the following phase, where attention vectors are created for each English word to show how closely connected they are to other words in the same phrase. How the learning mechanism work is that, every Greek given word will be translated into its English version based on the previous results. The result will then be compared to the actual English sentence that was fed into the decoder. After the comparison, the matrix value will be updated. This is how the model will learn after several iterations.

This block is called Masked as every next English word of our example, will be 'hidden' (or also called 'masked') at first and, without knowing the actual translated word, the model will independently predicts the following word using prior findings.

Again, an Add & Normalize step takes place.



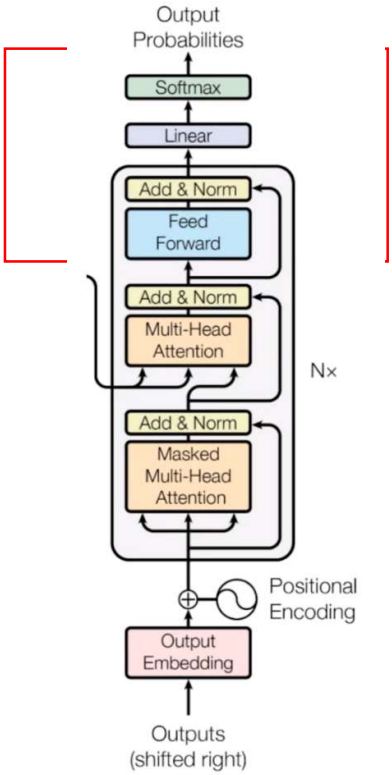
Picture 14: Masked Multi-Head Attention [4]



Picture 15: Multi-Head Attention [4]

Multi-Head Attention

Another Multi-Head Attention block receives the final attention vectors from the preceding layer and the vectors from the first encoder block. This is why this step is called Encoder-Decoder Attention block. Over this block, the mapping between each Greek and English word and the capture of their relationship within the given sentences is taking place. The output of this step is an attention vector for every word in both sentences. This step is sequential and is followed by an "Add & Normalize" layer.



Picture 16: Feed-Forward Network, Add & Norm, Linear and SoftMax [4]

Feed-Forward Network, Add & Norm, Linear and Soft-Max

The result of the second multi-headed attention is further processed in a pointwise feedforward layer. The result is then subjected to the final linear layer, which acts as a classifier and is as large as the number of classes in the challenge. A SoftMax layer processes the classifier's output and generates a probability score between 0 and 1. The anticipated word is shown by the index with the highest likelihood.

5.2 Transformers for Language Modeling

The Language modeling may be defined as the act of calculating the likelihood of the next word given the words that came before it. The whole corpus must be divided into standard-length parts with the purpose of train a Transformer model on textual data. The pre-trained model may be adjusted to carry out and resolve any job that is particular to a given situation. Modern models that have been trained on billions of words have been created after Transformers were introduced. Some of the pre-trained models have significantly enhanced several NLP jobs. State-of-the-art pre-trained transformer models have been developed, such BERT, XLNet, and RoBERTa, which will be discussed below.

5.2.1 Transfer Learning

As mentioned above, the idea of using pre-trained models describes the concept of transfer learning. These models were first trained using self-supervised learning methods on

sizable unlabeled text corpora [20]. Pre-training takes a lot of time and resources, and is often completed over several days using numerous GPUs. The datasets and learning goals used during pre-training vary greatly amongst models. These models' checkpoints can be then used as starting point for the fine-tuning of a specific NLP task where, usually a task-specific output layer needs to be attached to the model. Most of the transformers-based models that have been used for the purposes of this thesis are open source and their checkpoints were retrieved from Hugging Face [https://huggingface.co/].

Below, the models that have been selected, as they fit the subject of the text classification task will be described.

5.2.2 BERT

Google debuted the BERT model (Bidirectional Encoder Representations from Transformers) in 2018 [21]. A bi-directional Transformer called BERT was trained using 16GB of unlabeled text data. Masked Language Modelling and Next Sentence Prediction are the two tasks that BERT has been trained on, and solely employs the encoder to learn a latent representation of the input text. A series of tokens that have been embedded into vectors and processed by the NN make up the input text. The result is a series of vectors, each of which has the same index as an input character. BERT employs the Next Sequence Prediction and Masked Language Modeling as its two methodologies for defining the prediction objective.

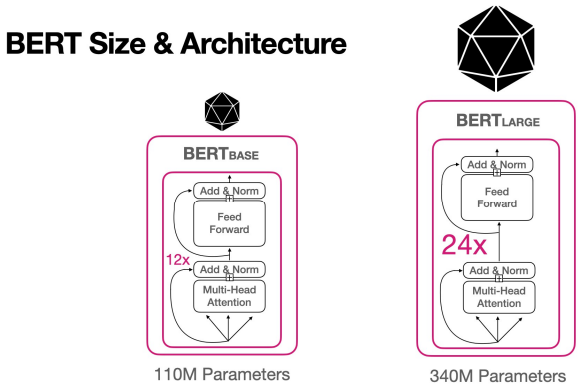
In Masked Language Modelling, a token (called a mask) is used in lieu of many words, and the model is trained to anticipate the missing word based on the context that the other non-masked words in the sequence provide. The encoder result must be combined with a classification layer, the output vectors must be multiplied by the embedding matrix to convert them into the vocabulary dimension, and then the probability of each word in the vocabulary must be determined using SoftMax.

Additionally, the model is trained to predict the subsequent sequence to a prior sequence in Next Sentence Prediction given two phrases. During training, 50% of the correct sentence pairs from the input are combined with 50% of the random sentence pairs to help BERT anticipate the next phrase more accurately. The input is processed in the following manner before being fed to the model: To assist the model distinguish between the two sentences during training, a [CLS] token is added at the start of the first sentence and a [SEP] token is added at the end of each sentence. Every token is embedded with a

sentence that denotes either Sentence A or Sentence B. Sentence embeddings and token embeddings with a vocabulary of two have a similar idea. To indicate its location in the sequence, each token is given a positional embedding. The following steps are done to ascertain whether the second assertion is indeed linked to the first. The whole input sequence is processed by the Transformer model. The result of the [CLS] token is converted into a 21-shaped vector using a simple classification layer, and the IsNextSequence probability is calculated using SoftMax. Masked LM and Next Sentence Prediction are simultaneously learnt while training the BERT model in order to decrease the combined loss function of the two methods.

With the process of fine-tuning, the pre-trained model is used to complete a variety of tasks such language translation, next sentence prediction, and text categorization. Sentiment analysis is categorized in the same way to Next Sentence classification by putting a classification layer on top of the Transformer result for the [CLS] token. When answering questions, a text sequence must contain the proper response, which the computer software must then indicate. Using BERT, a Q&A model may be trained by picking up two extra vectors that represent the start and end of the response. The objective of named entity recognition (NER) is to recognize the many categories of entities that could be present in a text sequence.

BERT Base, which has twelve layers, twelve attention heads, and 110 million parameters, and BERT Large, which has twenty-four layers, sixteen attention heads, and 340 million parameters, are two different versions of the model.



Picture 17: BERT Base & BERT Large Architecture [5]

The term Parameter describes the amount of the learnable variables that are available to each of the model. Transformer Layers refer to the number of the Transformers blocks,

which transform a sequence of word representation into a sequence of contextualized words, that are required. The Hidden Size determines the layers of mathematical function that are places between the input and the output and assign weights to the words in order to give the desire result. The Attention Heads determines the size of a Transformers block.

BERT's source code is publicly accessible, which lets users that focus their efforts on fine-tuning BERT to customize the model's performance to their unique tasks, while there are also thousands of open-source, pre-trained BERT models currently available for specific use cases if you somebody doesn't want to fine-tune BERT.

5.2.3 XLNet

Yang et al. [22] suggested an additional bi-directional Transformer in 2019. On 20 language challenges, XLNet outperforms BERT in terms of prediction accuracy thanks to its bigger data handling capacity and improved computational power. The output of the trained model in this auto-regressive model is the joint probability of a set of tokens built on the Transformers architecture with recurrence.

The XLNet architecture entails of a word-embedding matrix that gives each token in the vocabulary a fixed-length vector, turning the sequence into a set of vectors. Relating those tokens in a sequential manner is the next stage. The updated language model training aim, which learns conditional distributions for all permutations of tokens in a sequence, rather than the XLNet architecture, is what distinguishes the model from others.

The Transformer-XL is the foundation of XLNet, which was trained on 130 GB of text data. The model is trained using permutation language modeling, which predicts tokens in a random order. Stronger learning of bidirectional relationships results in better relationships between words. Both the XLNet-Base cased and the XLNet-Large cased versions of the network have twenty-four layers, sixteen attention heads, and 340 million parameters respectively.

5.2.4 RoBERTa

A substantially optimized BERT technique called RoBERTa [23] was presented by Yinhan Liu et al. The model is comparable to BERT, but it has removed the purpose of predicting the following phrase and has been trained for a longer period of time on more data with larger batches. In a word, RoBERTa entails data and input modification in addition to fine-tuning of the initial BERT model. It is a reimplementation of BERT with some

minor embedding adjustments and changes to the important hyperparameters. The RoBERTa model and the BERT model have the same architecture. This model uses 160GB of text for pre-training, with sixteen GB of Books Corpus and English Wikipedia used in BERT. In order to train the machine learning algorithm, dynamic masking is used to replace the Next Sentence Prediction task in BERT. In addition, the batch sizes are increased. This research shows that the RoBERTa model is more efficient than both BERT and XLNet models on GLUE benchmark results. The RoBERTa-Base model comprises 125 million parameters, twelve layers, and twelve attention heads. Twenty-four layers, sixteen attention heads, and 355 million parameters make up the RoBERTa-Large model.

5.2.5 XLM-RoBERTa

Unsupervised Cross-lingual Representation Learning at Scale is known as XLM-RoBERTa. and was proposed by Naman Goyal et al. [24]. It performs better in cross-lingual categorization than multilingual BERT because it was trained on a corpus of 100 languages called the Common Crawl Corpus. Utilizing just monolingual data, XLM-RoBERTa is trained with the multilingual MLM aim. The model was trained to anticipate masked tokens in the input, and samples of text streams are collected from each of the languages that were utilized during training. While BERT employs words or characters as the input in the model, XLM uses BPE, which enhances the shared vocabulary between various languages.

Each training sample in XLM has the identical text in both languages, and the model can predict tokens in the second language using the context from the first language.

There are several bidirectional transformer models that have already been trained. The variations were in how much data, how much computing power, or how a model was trained. There seems to be a trade-off among computation and prediction metrics. BERT is a good base line model for NLP tasks. The data is very diverse and it is time consuming to get familiar with it. In conclusion, what complicates the training process is the complexity and the volume of the data. Reducing the computational time, using fewer training data, whilst maintaining good performance is the key to achieving high performance.

5.2.6 FastBERT

In 2020, Liu et al., published a paper and proposed FastBERT [25]. The paper claimed that, as a fact, pre-trained language models, such as BERT, have demonstrated to be quite efficient. In several real-world situations, however, they are typically computationally

expensive due to the difficulty of executing such complicated models given the available resources. The paper introduced a speed tunable FastBERT with adaptive inference time to increase their effectiveness while ensuring model performance. The inference speed can be adjusted to match the needs of the problem, while avoiding unnecessary calculation. This model also employs a unique self-distillation approach for fine-tuning, which further promotes a greater computing effectiveness with less performance loss. Branches plus a backbone make up Fast-BERT. The branches have student-classifiers that are added to each Transformer output to enable early outputs, while the backbone is constructed on a 12-layer Transformer encoder with an extra teacher-classifier. In twelve datasets, including English and Chinese, this model achieved encouraging results. If alternative speedup criteria are applied to it, it can accelerate anywhere from 1 to 12 times faster than BERT.

6 Natural Language Processing

Natural language processing (NLP) is the field that lets a computer to understand and comprehend human-generated text and speech similarly to humans. Many studies have been made over this field over the years, which result into lots of different tools, frameworks, and techniques to use To comprehend the content, sentiment, and intent of an author as well as human languages in general, NLP employs machine learning, rule-based methods, and more recently, deep learning systems. NLP is used but not limited to activities such as, language machine translation, text classification, text summarization, chatbots, etc.

NLP has become increasingly important as a result of the growth of Big Data and their analysis. The huge volume of unstructured data (text data), the numerous sources that, in our days, produce them and the emersed need to analyze them are handled by NLP algorithms and techniques which help researchers, individuals and companies to automate the processes and make the understanding of data easy, in order to make better decisions. Social media data, online posts, blogs and virtual newspapers are important sources of data for public view. They can be used to mine information for insights into different topics. Therefore, businesses and individuals may benefit from employing NLP approaches to better understand their clients and goods.

To create reliable predictions, text preparation is required prior to the use of any machine learning algorithm and technique. This phase is essential since machine learning algorithms don't work with text materials that humans can read; instead, they work in a numeric space. Consequently, in order to analyze a text, we must process the text and turn it into vector representations called word embeddings. Data preprocessing, which eliminates noise and inconsistent data, and data vectorization, which uses word embeddings to represent the text, are the two categories into which the NLP procedures may be divided.

6.1 Data Preprocessing

We can reduce noise, low-level information, dimensionality and complexity of the problem and the dataset, by preprocessing the input text data with Data Preprocessing techniques. These techniques are not mandatory, and their use depends on the nature of the

problem and the algorithm and machine learning model that is about to be used. On the following section, common text preprocessing techniques that are usually used in NLP challenges will be described.

6.1.1 Tokenization

One of the most frequent jobs in NLP is tokenization. Tokens are called the small units that text is broken into. These tokens might be letters, words, or even sub-words. A dataset's unique tokens can subsequently be utilized to create the dataset's vocabulary, which will be made up of all the single tokens that were discovered on the data. One limitation of the above-mentioned technique is how it handles the Out-Of-Vocabulary (OOV) words. The algorithm will disregard a term if it isn't used in the problem's vocabulary and is given a brand-new example. The BPE [26] approach, a sub-word tokenization technique that can deal with the problem of OOV words, was developed to address this circumstance.

The predominant norm used for text tokenization today is a variation of BPE. Many algorithms, including the cutting-edge Transformers, utilise them [13]. A sub-word tokenization method from BERT based on BPE is called WordPiece [27]. WordPiece generates new words based on probability rather than frequency. Unigram-based [28] and SentencePiece [29] are two further variations. The latter is used to tokenize documents in languages where spaces are not used as word separators in sentences. In order to understand the structure of a sentence, it is helpful to use a sub-sequence of words extracted from it using a n-gram-based model.

6.1.2 Stop words' removal

In order to enhance the understanding of a text, an NLP technique is used known as word removal. This technique removes words that are not relevant, or that don't actually carry added value to the task at hand. This procedure purges redundant, noisy, and low-information terms from the input dataset, resulting in a more precise vocabulary. The most common words in a language, such as prepositions, pronouns, and conjunctions, are really those that are removed from the list since they appear in every document regardless of context. The exclusion of such words from a training set could lead to a decrease in the amount of tokens involved in the training process, since the training set is reduced in size and therefore the training time is improved.

Stop word removal as a task, is heavily dependent on the purpose of the objective. Sometimes, stop words can change the meaning of an entire sentence. This might be crucial during a sentiment analysis task on a dataset. For example, by removing the word ‘not’, the meaning of the negative sentiment of the entire sentence might change, making it more positive. Nowadays, especially when utilizing deep learning models like BERT, there are instances where the stop words give context for the author's meaning. Stop words attracted greater attention than non-stop words, as Qiao et al.[30] noted throughout their investigation.

6.1.3 Stemming & Lemmatization

Stemming and Lemmatization are both text normalization techniques in NLP that are used to re-form the words of the input text document into simpler tokens. What these techniques are trying to achieve is to reduce the form of the word into a simpler common base. By the use of these techniques, the size of the vocabulary is reduced significantly and the Machine Learning task is simplified. When the final few letters of a word are removed, a process known as stemming occurs, which usually leads to incorrect spelling and meanings. For instance, the letter "car" would result from stemming the word "caring." Lemmatization reduces a phrase to its logical fundamental form, or lemma, while taking context into consideration. For instance, lemmatizing the word "caring" would result in the term "care". Apparently, the process of lemmatization is much more time consuming compared to stemming, but in real-world problems stemmers end up to less reliable or effective results due to the fact that they just chop off suffixes [31]. Transformers on the other hand, which are considered to be the latest state-of-the-art techniques do not utilize none of these techniques. Transformers employ variations of the aforementioned BPE to condense the terminology.

6.1.4 Part-of-speech tagging

With part-of-speech (POS) tagging, words in a text are categorized in accordance with the specific part of speech to which they belong, based on the word's definition, and intended meaning. The structure of the lexical words in a text is described by POS tags. This knowledge may be used to create assumptions about the word meanings in the document. POS tagging can be utilized to a variety of tasks in NLP, such as text-to-speech, rearranging adjectives and nouns for translation, and enhancing syntactic parsing.

6.2 Vectorization

Certain machine learning algorithms can't process text documents in their native format, which is what humans are capable of. The algorithms need numeric representations of the text so as to do any task, such as classification, clustering, and regression. In order to solve machine learning problems, data vectorization is necessary. This will convert the input documents into word embeddings, which will make the system more efficient. Over the next subsection, common word embedding techniques which are used to produce word embeddings using statistical and prediction-based methods will be described.

6.2.1 One-Hot encoding

One-hot encoding is a method of creating word embeddings that is very simple and unsophisticated. Specifically, the vector represents categorical variables in the form of 0s and 1s. This algorithm is very fast but can also produce large and sparse matrices. The feature vectors grow as the vocabulary size grows, resulting in an inefficient word embedding process. The lack of semantic annotation in this sentence means that it is difficult to understand what the words mean. It is generally used as a reference model for comparisons.

6.2.2 Bag-of-words

Bag-of- Words (BOW) is a common technique for data vectorization. The document contains dummy variables that indicate if a specific word appears in it. The Document-Term Matrix (DTM) is a specialized encoding method that is used more often to encode words in documents than One-Hot Encoding. However BOW also generates binary vectors that are the same length as the vocabulary. This implies that the size of the embeddings grows together with the vocabulary size. Again, the words are dispersed over a wide region, making it harder to memorize them.

One-hot encoding and bag-of-words are not regarded as advanced techniques. By relying just on the frequency of the words in a particular document, these algorithms can provide extremely basic numerical representations of text. The algorithms' outputs are sparse, and they are typically utilized as the first stage in the development of more intricate and sophisticated techniques.

6.2.3 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TFI-DF) is a word embedding approach that assigns each word a weight based on how significant it is throughout the whole document and corpus. Measuring the frequency of each word in a document is the first stage in the process. The frequency with which a word appears in a document is measured using the term frequency (TF) metric. It is computed by dividing the amount of times a word appears in the text by its overall word count. The next step is to determine the data's document frequency. This gauges how frequently the term appears within the dataset's corpus. It specifically assesses the frequency and significance of each word throughout the whole corpus. It is computed by dividing the total number of documents by the proportion of documents that include the character X.

We can use TF-IDF to extract words that may be rare in the corpus. TF-IDF produces a DTM with the unique words of the corpus, similar to the BOW. The only variation is that the TF-IDF technique uses word occurrences frequencies within the document as well as a weight value to measure the importance of the word within the entire corpus. TF-IDF helps to penalize words that are most frequent in the corpus, and this can lead to a higher weight being given to words that may not have a high occurrence in the data.

Both methods mentioned have limitations because they are simple and do not consider the complexities of NLP. The algorithms don't consider the semantic meaning of the words in a document and the correlation between them, which can result in unsophisticated solutions.

6.2.4 Word2Vec

In 2013, Google researchers published a neural network for word embedding. This neural network is different from the previous methods for data vectorization, which used a Random Forest algorithm. Word2Vec [32] can obtain valuable information from text documents likewise a CNN extracts information from pictures. Word2Vec can preserve each word's semantic meaning and express it in the appropriate word embeddings. In order to demonstrate the semantic resemblance among the words, the output vectors are selected based on the cosine similarity function among the terms. It is possible to develop the approach using either a Continuous Bag of Words (CBOW) or a skip-gram.

CBOW refers to using context words as input in order to predict the output words. The order of the words does not affect the outcome of a sentence. Skip-gram uses the target

words to predict the context words which is exactly the opposite of CBOW. CBOW is more accurate than skip-gram and can represent rare words more efficiently than skip-gram. Both methods are useful for solving specific problems, but they need to be compared in order to see which one is the better choice for the specific problem.

To lower the cost function and fine-tune the weights, the Word2Vec neural network is trained utilizing backpropagation and stochastic gradient descent (SGD). Because comparable words in a corpus will have similar vector embeddings and dissimilar words will be further away in the dimensional space, the Word2Vec methodology is superior to simpler statistical methods.

6.2.5 GloVe

GloVe is a data vectorization tool that was built by researchers at Stanford University in 2014 [33]. It takes an approach to data vectorization that is different from other tools out there. GloVe is a model that uses a weighted least squares objective to calculate the results. The algorithm's fundamental premise is that it can use the ratios of the global word-to-word co-occurrence probabilities to encode some form of meaning. The performance of GloVe increases as long as there are negative samples chosen. Even though the GloVe algorithm is based on a word-context matrix and the training of the algorithm yields a large matrix of co-occurrence information, in general, GloVe performs better than Word2Vec.

6.2.6 FastText

FastText is an extension to the Word2Vec library that was developed by Facebook in 2016 [34]. It specifically skip-gram data so that it can be more efficiently processed. Word2Vec treats each word as a single entity, while FastText treats each word as a vector of characters. FastText uses an algorithm to combine the n-grams of the words in a text to create a final word. The big benefit of FastText is that it produces better word embeddings for rare and OOV words.

All of the approaches outlined above, however, fail to produce word vectors that include information on the arrangement of the words in a particular phrase and their relationship to one another in regard to their placement. The problem with a word's placement in a sentence and the context surrounding it is that the information carries over to the next sentence. The meaning of the author's aim may change if a word is shifted from the beginning to the conclusion of a sentence.

Transformers were introduced as an architecture containing of an encoder and a decoder. According to Vaswani et al. [13], unlike the sequence-to-sequence models like LSTM and GRU units, Transformers' architecture, named after the article "Attention Is All You Need," is a sequence transduction model that is solely based on the attention mechanism, substituting multi-headed self-attention for the recurrent layers seen in an encoder-decoder architecture. By avoiding recursion, parallel computing is made possible and performance loss caused by the extremely lengthy dependencies of the text data is minimized. Transformers are able to parallelize the computations and produce cutting-edge results as opposed to the sequential method used by recurrent Neural Networks (RNN). The benefit of the Transformers is that they can simultaneously use the location information of the words and contain all of the context of the words in a given corpus.

The use of text processing and machine learning techniques in the field of natural language processing has seen a major change in recent years. This has led to increased efficiency and accuracy in these fields, making them increasingly important in today's world. With our understanding of the optimum way to convert words into vector representations and maintain the word-to-word linkages with the semantic meaning, a new era has begun. This allowed to better understand how words are used and how they work together to create meaning. The NLP research community has developed powerful models that are better than other machine learning methods in many tasks related to NLP. To tackle a wide variety of challenging NLP tasks, many designs based on the Transformer were developed. The GPT-3 [35], BERT [21], XLNET [22], and RoBERTa [23] are a few of the most popular models that have been developed.

Because deep neural networks are more widely used and computer resources are more plentiful, we can process and analyze massive amounts of data more quickly and effectively than ever before. The recent technological advancements have made computers much more accurate and faster, which has led to a widespread adoption of NLP by commercial companies and individual researchers. Most of the NLP investments have been made by large technology companies. Many companies offer Application Programming Interface (API)-based NLP solutions that allow for pre-trained models of large corpora. Pre-trained models have the tremendous benefit of being reusable and fine-tunable, which means that we only need to train them one time before someone may use them and adjust them to solve their particular problem.

6.3 Data Augmentation

Without accurate data, it is impossible to create an accurate model and get a reliable outcome. Regardless of if it is a classification or regression problem, numerous steps are required prior to initializing a model. An exploratory analysis may help identify any patterns or irregularities in data. Data imbalance is one of the most common challenges with real-world data. This can be caused by the data being too scattered or incomplete, or by the data being too similar or different. The problem with having too much or too little data in different classes is that it can lead to imbalance in the data. This essential problem makes generalization challenging. Algorithms discover patterns or gain knowledge from their errors. On the same manner, a real-world problem may require solution, but the input data might not be enough.

6.3.1 Text Data Augmentation

A common technique in order to enhance model's performance is Data Augmentation. This entails creating new synthetic data from existing data, modifying existing ones or converting them. Tokenizing documents into a phrase, rearranging and rejoining them to create new texts, or substituting adjectives and verbs with their equivalents to create various texts with the same meaning are all examples of text augmentation techniques. An embedding is a data structure that provides a way to embed a text representation of a set of words into another set of data. There are several word embedding text representations that exist.

In order to create new sentences and words, comparable words and phrases can be swapped out. Another strategy that is based on text translation can likewise provide fresh synthetic data. Converting between languages is an intriguing task. This will create new translations of the data in the database. A translator will translate the target language into another language and then back to the target language.

Because the grammatical and syntactic structure of the text dictates its meaning, data augmentation in NLP activities should be done with caution. Before the training phase, text generation techniques are used. Prior to training the model, a fresh, enhanced dataset is created and fed into data loaders.

Another approach suggests using an algorithm to identify sentence fragments that appear in similar environments, proposing that swapping such fragments with others should create similar outcome. Other approaches (also known as 'Easy Data Augmentation',

introduced by Wei and Zou in 2019 [36]), are Synonym replacement, Random Insertion, Random Swap and Random Deletion.

7 Web Content Classification Analysis

Over this chapter of the thesis, the purpose of this exercise, along with the methodology and the experiments of the web content classification analysis will be described. This case study aims to classify as accurate as possible the web page URLs of a given dataset, given their content, and hence improve analytics, the insights and therefore the data decision-making approaches of the stakeholders.

Techniques about data preprocessing, data augmentation and data modeling as well as the results of the models will be described.

7.1 Case study

In this thesis, the main focus is around a company which, during the construction of the website, not all URLs were tagged. This issue was identified to be a bottleneck for their website analytics. The process of updating the metadata field after hand, requires a page to be deleted and recreated. This approach requires a lot of manual annotation and effort and will create many issues over the downstream analytics solutions of the particular company. For the above-mentioned reasons, what is suggested is to classify all the URLs based on their content and implement the updated metadata right after the data collection, over the data base which will then be used as a source for every associated downstream solution in order to maintain consistency as well as to oversee results.

7.2 Dataset

The initial dataset included all the URLs of a particular website of company, which were publicly available and accessible. The training dataset was filtered based the URL status code. A server's answer to a browser request is a Hypertext Transfer Protocol (HTTP) status code. When a user accesses a website, their browser makes a request to the server of that website, and the server replies with a three-digit code called the HTTP status code. This process was made with the Requests library [37]. Only the accessible sites were passed into the next step in order to extract their text content.

A web scraping approach (using Trafiltura library [38]) was applied in order to extract each URL's content, focusing only on the useful text content (ignoring footer, header, images, hyperlinks etc., which would not add any value on this NLP Classification task). Alongside, some particular metadata were gathered, as those meant to be the class of each URL. The quantity of the unique classes of the dataset was 19. Over the 1st iteration, 86.58% of the dataset is tagged as 'UNKNOWN'. The aim of this exercise was to eliminate as much as possible the amount of the 'UNKOWN' tags and classify each URL to one of the 19 classes. The correctly classified 13.42% of the dataset is being used, while the rest will be used as test, proving the accuracy over self-annotated test which require opening multiple URLs and validating each class. This task is performed from an individual with domain knowledge.

7.3 Dataset preprocessing

One of the most essential/fundamental steps of the ML pipelines is preprocessing. It is the process of converting the raw input data into a format that is appropriate and more normalized for the algorithms. For example, noise and grammatical mistakes are frequently present in the raw data. These mistakes have no contribution to the sense of the statement that would be helpful.

In order to enhance the training dataset as mentioned above, one of the preprocessing steps that has taken place included a parse of the content of the accessible URLs that were classified as UNKOWN in order to see if any of the name of the classes was included. The classes in this particular case represent products of the company, so if any of them exists within the content of a page URL or even in the URL itself, indicates it's class. This process enhanced the training dataset with +13% more classified page URLs that was used as an input for the training of the classification model.

Furthermore, any special characters like #, @, :, \, _ , company details and any menu/ button content were removed. Additionally, any extra spaces that were left by the processing methods or the original data were removed. On top of that, all the words were lower-cased in order to eliminate as much as possible the number of the unique tokens.

7.4 Baseline models

The first step of the exercise was to build some initial baseline models, without any further pre-processing on the training dataset neither any parameter fine-tuning. The results of

these models gave a first hint of the results of the study and were also used as a reference point for the next steps.

The training dataset was split into train/test (0.75, 0.25) in order to iterate as a first step of validation of the model [39].

The models that were trained were bert-base-uncased, distilbert-base-uncased, albert-base-v2, roberta-base and fast-bert.

A pretrained model on the English language employing a Masked Language Modeling (MLM) aim is called **Bert-base-uncased**. The dataset of 11,038 unpublished books from BookCorpus and English Wikipedia served as the pretraining data for the BERT model (excluding lists, tables and headers). BERT, as mentioned over the previous chapters, was introduced by Devlin et al., in 2019 [21]. There are various model variations and this particular model is uncased which means that it does not make a difference between capitalized and non-capitalized words. An auto-tokenizer from the pre-trained bert-base-uncased was used. Tokenizers will first separate a given text into tokens, which are often words (or parts of words, punctuation marks, etc.). In order to create a tensor out of those tokens and give it to the model, it will next turn those tokens into integers. It will also include any extra inputs that the model would need in order to function properly. The padding control argument was set to "max lenght" to pad to either the length given by the max length argument or the longest length that the model will tolerate. No “max_lenght” input was provided, so the padding was default into the maximum length the model can accept. The argument truncation was set as ‘True’ which means that only the first sentence of a pair if a pair of sequence (or a batch of pairs of sequences) is provided will be truncated. Using ‘bert-base-uncased’ from pretrained also loads the model weights. The model was trained for 15 epochs and achieved an 0.86 accuracy. A given sentence was then classified correctly to its class with a 0.67 score over the rest of the classes.

A distilled version of the BERT basic model is the **Distilbert-base-uncased** model. Distil-BERT is a transformers model that was pretrained on the same corpus in a self-supervised manner utilizing the BERT base model as a teacher. It is smaller and quicker than BERT. The model was taught to produce the same probabilities as the BERT basic model because it was pretrained with the goal of distillation loss. It was also pre-trained with the MLM aim, thus when given a phrase, the model randomly masks 15% of the input words before processing the complete masked text and asking the user to guess the words that were hidden. n contrast, traditional RNNs often see the words in order. As a

result, the model may discover a two-way representation of the statement. The cosine embedding loss indicates that the model was trained to produce hidden states that were as comparable to the BERT base model as was practical. By doing so, the model acquires the same internal representation of the English language as its instructor model while performing downstream or inference tasks more quickly. An auto-tokenizer from the pre-trained distilbert-base-uncased was used with the exact same parameters as the previous model. Using 'distilbert-base-uncased' from pretrained also loads the model weights. The model was trained for 15 epochs and the achieved an 0.9 accuracy. A given sentence was then classified correctly to its class with a 0.7 score over the rest of the classes.

Albert-base-v2 is an ALBERT [40] model which was introduced from Lan et al. in 2019, is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. The model was pretrained with two objectives, the MLM, which was earlier described and the Sentence Ordering Prediction (SOP). ALBERT employs a pre-training loss based on foreseeing the arrangement of two subsequent text segments. If the dataset includes of labeled sentences, a conventional classifier may be trained utilizing the features produced by the ALBERT model as inputs. With this method, the model acquires an internal representation of the English language from which properties beneficial to later applications may be extracted.

ALBERT is unique in that its Transformer shares its layers. As a result, the weights of all layers are equal. The computational cost is identical to a BERT-like design with the same number of hidden levels even when repeating layers are used since the equal number of layers must be iterated over. An Albert-tokenizer from the pre-trained albert-base-v2 was used with the exact same parameters as the previous model. Using 'albert-base-v2' from pretrained also loads the model weights. The model was trained for 15 epochs and the achieved an 0.67 accuracy. A given sentence was then classified correctly to its class with a 0.6 score over the rest of the classes.

roberta-base is a RoBERTa model which is a transformers model pretrained on a big corpus of English data in a self-supervised fashion. The model was pretrained with the MLM objective, which was described above. An RobertaTokenizer from the pre-trained roberta-base was used with the exact same parameters as the previous model. Using 'roberta-base' from pretrained also loads the model weights. The model was trained for 15 epochs and the achieved an 0.89 accuracy. A given sentence was then classified correctly to its class with a 0.7 score over the rest of the classes.

The development of a DataBunch object is necessary for **fast-bert** since it transforms the data into internal representation for BERT, RoBERTa, DistilBERT, or XLNet. Additionally, based on the device profile, batch size, and max sequence length, the object instantiates the appropriate data-loaders.The DataBunch object used a bert-base-uncased tokenizer. The model also requires the creation of a Learner object which will take the DataBunch created earlier as input along with some of the other parameters. The most important hyperparameter for this model is the optimal learning rate (for the purpose of this exercise, the learning rate finder proposed by Leslie Smit was used [41]). The model was trained for 15 epochs and the achieved an 0.91 accuracy. A given sentence was then classified correctly to its class with a 0.8 score over the rest of the classes.

7.5 Data augmentation & model training

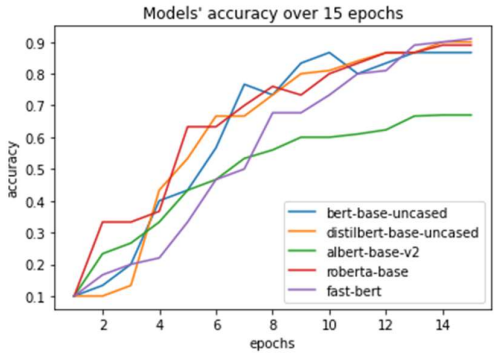
As mentioned over the previous chapters, a data augmentation technique was used. For the purposes of this exercise, which required text augmentation, nlpaug [42] was used. A library for textual enhancement in machine learning experiments is called nlpgaug. By producing textual data, the performance of the deep learning model is to be improved. Character level augmentation, word level augmentation, and flow/sentence level augmentation are the three types of augmentation that the nlpaug offers. For the purposes of this exercise, the word level augmentation was used. With this approach, words will be replaced within a sentence based on substitution. Using this data augmentation approach, the dataset became balanced by creating substitute sentences for each class until all classes are equally represented.

The previously baseline trained models have now been re-trained over the enhanced dataset after the data augmentation to see if this technique improved the results. The number of the epochs for each model were also increased to 25. The same sentence was also tested on each model and bert-base-uncased achieved the highest score for the correct prediction of the class with 0.99.

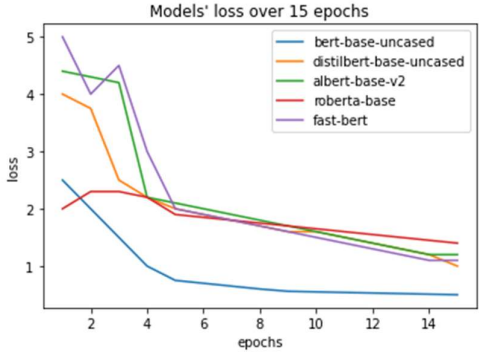
7.6 Results

With this exercise, we experimented with the problem of the non-classified web data, and in particular the non-properly tagged URL pages of a website of a company, which has an impact on the accurate analysis of the data. In order to avoid the manual classification of these webpages and also the reconstruction of the web page, the web content

classification approach is proposed. Having performed the required pre-processing on the data, the models were trained to be used it as a baseline.

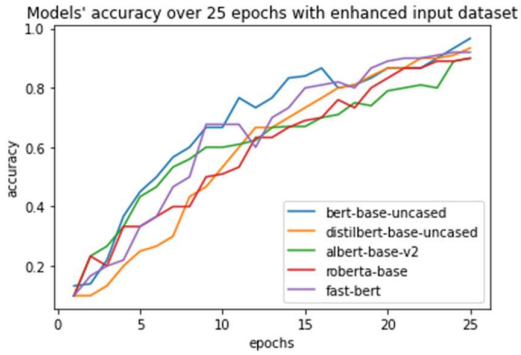


Picture 18: Line graph of validation accuracy over the number of epochs

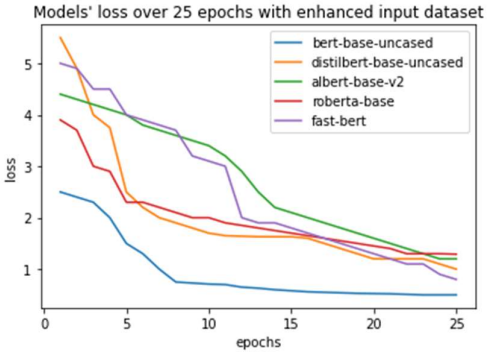


Picture 19: Line graph of validation loss over the number of epochs

Since the dataset was imbalanced, and also in order to enhance the performance of the models, a data augmentation technique was used in order to create equally balanced categories, which will help all classes be equally represented for a better prediction of them and the elimination of the bias. After proceeding with the data augmentation, a new round training was set and the results will be presented.



Picture 20: Line graph of validation accuracy over the number of epochs



Picture 21: Line graph of validation loss over the number of epochs

All of the models were trained, validated, and tested on the same dataset.

The total performance has been evaluated using the accuracy score, as the prediction of the True positives and the True negatives is more suitable for the purposes of this exercise.

	Epochs	Accuracy (base line)	Epochs	Accuracy (with augmented text)
bert-base-uncased	15	0.86	25	0.967
distilbert-base-uncased		0.90		0.934
albert-base-v2		0.67		0.90
roberta-base		0.89		0.90
fast-bert		0.91		0.92

Table 1: Accuracy results (*table created by the author*)

The above table indicates that even though that the baseline models achieved a satisfactory accuracy score, by enhancing the dataset and increasing the number of the training epochs the results have been drastically improved.

Based on the results of the previous pictures, bert-base-uncased was proven to achieve the highest accuracy score and the lowest loss, which indicates the difference from the desired target state(s). One possible reason for this superior performance is that BERT is a transformer-based model, which has been shown to be highly effective for NLP tasks due to its ability to process long-range dependencies in language. Additionally, BERT was specifically designed for pretraining on large amounts of unannotated text, which likely gave it an advantage in this task.

On the other hand, the other models we compared had lower accuracy. For example, DistilBERT is a smaller, distilled version of BERT, which may have reduced its ability to accurately classify text. Similarly, ALBERT and RoBERTa are also based on transformers, but they use different training techniques and have different architectural choices, which may have negatively impacted their performance. Finally, FastBERT is an optimization of BERT that is designed to be faster to train, but this optimization may have come at the cost of accuracy.

Overall, our results suggest that BERT-base-uncased is a strong choice for text classification tasks, likely due to its transformer architecture, pretraining on large amounts of unannotated text and ability to capture contextual information from the entire input sequence. While other models may be faster or more resource-efficient, they did not achieve the same level of accuracy in our study.

Noteworthy is the fact that the relative performance of these models may vary depending on the specific dataset and task. For example, RoBERTa has been shown to perform well

on some tasks, such as language translation, where it outperforms BERT. However, in the context of our text classification task, BERT was the clear winner.

While other models may have different strengths and may be more suitable for certain tasks, they did not achieve the same level of accuracy as BERT in our study.

To compare performance with previously labelled data, which in this case were initially tagged as 'UNKNOWN', 40 entries were selected and fed into the pipeline for prediction. Relying on the domain knowledge, we can validate that all of them were correctly classified to their corresponding class.

7.7 Conclusions & Future Work

The conclusion of this exercise if that a company could rely on the results of a Transformers model for the improvement of their metadata and by following a similar approach one can avoid spending additional resources. Modern NLP classification approaches, especially when referring to Transformers' related approaches provide high performance outcomes which can be leverage for any required aspect on a real-life problem.

Due to a lack of training resources, the exercise's results could have been more successful. These capabilities could be increased, the chosen model could continue to be fine-tuned, training epochs and batch sizes could be increased, and the model could be retrained using a dictionary or corpus that is relevant to the exercise's domain. Additionally, the outcomes of this classification exercise can be utilized as a POC to start and continue a similar classification operation for additional company-owned assets.

Bibliography.

[1]Joseph, Sethunya & Sedimo, Kutlwano & Kaniwa, Freeson & Hlomani, Hlomani & Letsholo, Keletso. (2016). Natural Language Processing: A Review. Natural Language Processing: A Review. 6. 207-210.

link:https://www.researchgate.net/publication/309210149_Natural_Language_Processing_A_Review#:~:text=Natural%20Language%20Processing%20(NLP)%20is,beings%20understand%20and%20use%20language.

[2]Zhao, Bo. (2017). Web Scraping. 10.1007/978-3-319-32001-4_483-1.

link:https://www.researchgate.net/publication/317177787_Web_Scraping#:~:text=Web%20scraping%20or%20web%20crawling,from%20text%20such%20as%20HTML.

[3]Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. ACM Comput. Surv., 34(1), 1–47. http://doi.org/10.1145/505282.505283

link:http://nmis.isti.cnr.it/sebastiani/Publications/ACMCS02.pdf

[4]Deepshikha Patel, Dr. Prashant Kumar Singh, (2013). A Web Page Classification technique with Textual Content Analysis Using NN-PCA for Objectionable Web Page Classification. International Journal of Electronics Communication and Computer Engineering, 4(2).

link:https://ijecce.org/administrator/components/com_jresearch/files/publications/IJECCE_1513 Final.pdf

[5]Inma Hernández, Carlos R. Rivero, David Ruiz, Rafael Corchuelo, (2014). CALA: An unsupervised URL-based web page classification system, ISSN 0950-7051,

link:https://doi.org/10.1016/j.knosys.2013.12.019

[6]Asheghi, Noushin & Markert, Katja & Sharoff, Serge. (2014). Semi-supervised Graph-based Genre Classification for Web Pages. 39-47. 10.3115/v1/W14-3706.

link:https://aclanthology.org/W14-3706.pdf

[7]Kathirvalavakumar Thangairulappan, Aruna Devi Kanagavel (2016). Improved Term Weighting Technique for Automatic Web Page Classification DOI: 10.4236/jilsa.2016.84006

link:https://www.scirp.org/(S(i43dyn45teexjx455qlt3d2q))/journal/paperinformation.aspx?paperid=70875

[8]Soner Kiziloluk, Ahmet Bedri Ozer (2017) Web Pages Classification with Parliamentary Optimization Algorithm. International Journal of Software Engineering and Knowledge Engineering VOL. 27, NO. 03

link:https://doi.org/10.1142/S0218194017500188

[9]Fahri Aydos, A. Murat Ozbayoglu, Yahya Sirin,, M. Fatih Demircia, (2020). Web page classification with Google Image Search results

link:https://arxiv.org/pdf/2006.00226.pdf

[10]Amit Gupta, Rajesh Bhatia (2021) Knowledge Based Deep Inception Model for Web Page Classification. Journal of Web Engineering

link:https://journals.riverpublishers.com/index.php/JWE/article/view/6657

[11]Kurt, M. S. & Yücel Demirel, E. (2022). Web Page Classification with Deep Learning Methods. Uludağ Üniversitesi Mühendislik Fakültesi Dergisi , 27 (1) , 191-204. DOI: 10.17482/uumfd.891038

link:https://dergipark.org.tr/en/download/article-file/1617495

[12]Yu Y. (2022). Web Page Classification Algorithm Based on Deep Learning. Computational intelligence and neuroscience, 2022, 9534918.

link:https://doi.org/10.1155/2022/9534918

[13]Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems. 5998–6008.

link:https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[14]Frank Emmert-Streib and Matthias Dehmer, Evaluation of Regression Models: Model Assessment, Model Selection and Generalization Error, 2019

link:https://www.researchgate.net/publication/345473993_Evaluation_of_Regression_Models_Model_Assessment_Model_Selection_and_Generalization_Error

[15]Margherita Grandini, Enrico Bagli, Giorgio Visani, Metrics For Multi-Class Classification: An Overview, 2020

link:https://arxiv.org/pdf/2008.05756.pdf

[16]Juri Opitz and Sebastian Burst. Macro F1 and Macro F1. 2019. arXiv: 1911.03347 [cs. LG]

link:https://arxiv.org/abs/1911.03347

[17]Ramón Quiza and J. Paulo Davim, Computational methods and optimization, 2011
link:https://www.researchgate.net/publication/234055177_Computational_Methods_and_Optimization

[18]Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun, Deep Residual Learning for Image Recognition, 2005

link: https://arxiv.org/pdf/1512.03385v1.pdf

[19]Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton, Layer Normalization, 2016

link: https://arxiv.org/pdf/1607.06450v1.pdf

[20]Matthias Aßenmacher, Christian Heumann, On The Comparability Of Pre-Trained Language Models, 2020

link:https://arxiv.org/pdf/2001.00781.pdf

[21]Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pretraining of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018).

link:https://arxiv.org/abs/1810.04805

[22]Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems 32 (2019).
link:https://papers.nips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html

[23]Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019).
link:https://arxiv.org/abs/1907.11692

[24]Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale.
link:https://aclanthology.org/2020.acl-main.747/
[25]Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, Qi Ju. FastBERT: a Self-distilling BERT with Adaptive Inference Time (2020)
link:https://arxiv.org/abs/2004.02178
[26]Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909 (2015).
link:https://arxiv.org/abs/1508.07909
[27]Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, Denny Zhou. Fast WordPiece Tokenization (2020)
link:https://arxiv.org/abs/2012.15524
[28]Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. arXiv preprint arXiv:1804.10959 (2018).
link:https://arxiv.org/abs/1804.10959
[29]Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv:1808.06226 (2018).
link:https://arxiv.org/abs/1808.06226
[30]Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. arXiv preprint arXiv:1904.07531 (2019).
link:https://arxiv.org/abs/1904.07531
[31]Vimala Balakrishnan and Ethel Lloyd-Yemoh. 2014. Stemming and lemmatization: a comparison of retrieval performances. (2014).
link:http://www.lnse.org/papers/134-I3007.pdf
[32]Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Effcient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013).
link:https://arxiv.org/abs/1301.3781
[33]Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 1532–1543.
link:https://nlp.stanford.edu/pubs/glove.pdf
[34]Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics 5 (2017), 135–146.
link:https://arxiv.org/abs/1607.04606
[35]Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020).
link:https://arxiv.org/abs/2005.14165
[36]Wei, Jason and Kai Zou (Nov. 2019). “EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks”. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, pp. 6383– 6389

link:https://www.aclweb.org/anthology/D19-1670
[37]Requests: HTTP for Humans, Release v2.28.1.
link:https://requests.readthedocs.io/en/latest/
[38]Barbaresi, A. “Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction”, in Proceedings of ACL/IJCNLP 2021: System Demonstrations, 2021, p. 122-131. DOI: 10.18653/v1/2021.acl-demo.15
link:https://trafilatura.readthedocs.io/en/latest/used-by.html
[39]Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
link:https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[40]Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations (2019)
link:https://arxiv.org/abs/1909.11942
[41]Leslie N. Smith , Cyclical Learning Rates for Training Neural Networks. (2015)
link:https://arxiv.org/abs/1506.01186
[42] nlpaug librbary
link

Figures

[1]Ravish Kumar, Supervised, Unsupervised and Semi-supervised Learning with Real-life Usecase (visited on December 2022)

link:https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning

[2]Ravish Kumar, Classification and Regression Problems in Machine Learning (visited on December 2022)

link:https://www.enjoyalgorithms.com/blogs/classification-and-regression-in-machine-learning

[3] Margherita Grandini, Enrico Bagli and Giorgio Visani, Metrics for multi-class classification: an overview, 2020

link:https://arxiv.org/pdf/2008.05756.pdf

[4] Lena Voita, NLP Course, 2022

link:https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

[5] Briteny Muller, BERT 101 State of the art NLP model explained, 2022

link:https://huggingface.co/blog/bert-101