Athens University of Economics & Business

Department of Management Science & Technology

MSc in Business Analytics

Machine Learning & Content Analytics

"Argument Detection Project"

MSc Student_1:	MSc Student_2:	MSc Student_3:
Agapiou Marios	Kontos Christos	Skarlis Vasileios
A.M.: f2821901	A.M.: f2821906	A.M.: f2821912

Table of Contents

Ensure healthy lives and promote well-being for all at all ages – Argument detection using deep Neural Networks

Abstract

The aim of this project is to accurately classify and predict text content and specifically categorize sentences with claims and evidences by its relevance to the measurable indicators of the United Nations' Sustainable Development Goal #3 (Good Health and Well-Being) using a variety of Deep Neural Network models for multiclass text classification, that have been demonstrated through the years to be capable of achieving remarkable performance in sentence and document modelling. This approach will help governments to take better decisions and make more accurate actions concerning the elimination of the specific problem, which has plagued for many years millions of people. We will consider three network models: the first one is a Feed Forward Neural Network (MLP), the second is a Convolutional Neural Network (CNN) and the third one is a Long Short-Term Memory Recurrent Neural Network (LSTM). We will take into account several open access records and scientific articles collections / documents from PubMed and we will investigate on the behavior of the neural networks by defining which model give us the best results and subsequently evaluate the predictions of the best model using the PubMed scientific articles collection as test case.

1. Project Description

Ensuring healthy lives and promoting well-being for all at all ages is important to building prosperous societies. Health for all people, all over the world, is an important part of sustainable development. However, despite great strides in improving people's health and well being in recent years, inequalities in health care access still persist. More than six million children still die before their fifth birthday each year, and only half of all women in developing regions have access to the health care they need. Various diseases exist that cause serious health issues, including epidemics like tuberculosis, HIV/AIDS, polio and malaria, where fear and discrimination limit people's ability to receive the services they need to live healthy and productive lives. Access to good health and well-being is a human right, and that is why the Sustainable Development Agenda offers a new chance to ensure that everyone can access the highest standards of health and health care— not just the wealthiest.

The Sustainable Development Goals (SDGs) or Global Goals are a collection of 17 interlinked goals designed to be a "blueprint to achieve a better and sustainable future for all". The SDGs, set in 2015 by the United Nations General Assembly and intended to be achieved by the year 2030, are part of a UN Resolution called the "2030 Agenda". Between the many resolutions, speeches, reports and other documents that are produced each year, the United Nations is awash in text. It is an ongoing challenge to create a coherent and useful picture of this corpus. In particular, there is an interest in measuring how the work of the United Nations system aligns with the Sustainable Development Goals (SDGs). In particular, there is a need for a scalable, objective and consistent way to measure and classify the main arguments that are being detected in publications concerning the specific SDG target, with the ultimate goal being the insurance of healthy lives and the promotion of well-being for all at all ages.

With the increasing availability of electronic documents and the extraction of open data from any source and form, the task of automatic categorization of documents became the key method for organizing the information and knowledge discovery. Their proper classification and categorization, need text mining, machine learning and natural language processing techniques to get meaningful knowledge. In general, text classification plays an important role in information extraction and summarization, text retrieval, and question-answering. Nowadays, deep learning based models have surpassed classical machine learning based approaches in various text classification tasks, including sentiment analysis, news categorization, question answering and natural language inference.

Therefore, the purpose of this project is to classify text content by its relevance and more specifically to accurately predict sentences including claims and evidences of various documents to the measurable indicators of the United Nations' Sustainable Development Goal #3 – Health and Well-Being (SDG3) using several Deep Neural Networks for label and multi-class text classification in order to help us identify patterns and gaining important insights, by eliminating as much as possible the time and the volume of documents one's has to read. Specifically, we developed three network models for our calculations, a Feed Forward Neural Network (MLP model), a Convolutional Neural Network (CNN model) and a Long Short-Term Memory Recurrent Neural Network (LSTM or RNN model), using data derived from PubMed open access records and scientific articles collections, filtered for studies using targets and indicators of the specific SDG.

This project is organized as follows. Following this description, the project discusses the objectives and the necessary business workflows in which our project is integrated. In the second section is extensively analyzed previous works concerning the specific approach as well as our way of approaching the problem. The third section explains all the procedures of gaining *our data, the preprocess methods that we follow in order to bring them into a remarkable form to develop our algorithms properly, as well as important statistic results arising from the data used in this project. The fourth section presents the methodologies and architectures that we followed for the development of our neural network models as well as all the parameters, hyper-parameters, evaluation measures and optimizers that we used. A fifth section presents also all the results that emerged from our analysis as well as the selection and evaluation of the best model, by including also all the necessary information gained from the predictions and the important metrics (learning curves, confusion matrices, classification reports) emerged from the test case we chose to proceed. The last section concludes with suggested areas for future work and the main conclusions drawn from the work of the specific project.*

i) Objectives and contribution of this project

The United Nations is a source of big data in the form of text. Between the many resolutions, speeches, meetings, conferences, studies, reports and internal regulations that exist and that are produced each year, the UN is awash in text. Even in a single department of the United Nations Secretariat, the amount of publications is significant. In the Department of Economic and Social Affairs (DESA), publications are central to its overall mission to support international cooperation in the pursuit of sustainable development for all. They inform development policies, global standards and norms on a wide range of development issues that affect peoples' lives and livelihoods.

However, very few people are in a position to see much more than a small sliver of specialized text. Even fewer can parse the various streams into a coherent and useful picture. What is needed is a quick and objective way to analyze large quantities of United Nations publications according to a desired criteria, namely the Sustainable Development Goals (SDGs). Each SDG has several "targets", or social outcomes that the UN hopes to achieve by 2030. Each of these targets is measured using a set of indicators. These indicators represent the quantitative measurements that will be used to judge whether each SDG target has been achieved or not by 2030. Specifically, SDG3 has 14 targets and 27 indicators.

Our challenge was to gather as much as possible information from all these publications and articles referring to the specific SDG's targets and indicators, in order to predict and classify properly, claims and evidences that arise from all these different publications contained inside the PubMed open access records and scientific articles collections library. By doing this and using appropriate neural network models, we succeeded to reduce the extra noise appeared in all of these documents and successfully isolated only the important information needed, in order to get more specific insights later with the right categorization of claims and evidences drawn from the specific SDG's targets and indicators.

Using machine learning models and especially deep neural network algorithms to analyze digital texts has many advantages. Over the past several years, the global Data Science community has watched the rise and steady penetration of such concepts as neural networks, Deep Learning, and back propagation. Algorithms can be used at scale with objectivity and can help identify patterns across publications and over time. This approach can also serve as a tool to explore and discover new texts and to inform the direction of future research. More importantly, this method hopefully will inspires other efforts to use modern data analytics to better understand the body of work of the United Nations. Finally, we believe that this method will help institutions as well as governments to take better actions and decisions concerning this SDG, in order to confront this problem and possibly to constitutes the cornerstone of the elimination of this problem that has plagued for years millions or even billions of people.

2. Mission

Text classification, also known as text categorization, is a classical problem in natural language processing (NLP), which aims to assign labels or tags to textual units such as sentences, queries and documents. Machine learning models have drawn a lot of attention in recent years. Most classical machine learning based models follow the popular two-step procedure, where in the first step some hand-crafted features are extracted from the documents (or any other textual unit), and in the second step those features are fed to a classifier to make a prediction. Some of the popular hand-crafted features include bag of words (BoW), and their extensions.

Popular choices of classification algorithms include Naïve Bayes, support vector machines (SVM), hidden Markov model (HMM), gradient boosting trees, and random forests. The two-step approaches have several limitations. For example, reliance on the hand-crafted features requires tedious feature engineering and analysis to obtain a good performance. In addition, the strong dependence on domain knowledge for designing features makes the method difficult to easily generalize to new tasks. Finally, these models cannot take full advantage of large amounts of training data because the features (or feature templates) are pre-defined.

A paradigm shift started occurring in 2012, when a deep learning based model, AlexNet [1], won the ImageNet competition by a large margin. Since then, deep learning models have been applied to a wide range of tasks in computer vision and NLP, improving the state-of-the-art [2–5]. These models try to learn feature representations and perform classification (or regression), in an end-to-end fashion. They not only have the ability to uncover hidden patterns in data, but also are much more transferable from one application to another. Not surprisingly, these models are becoming the mainstream framework for various text classification and argumentation mining tasks in recent years.

Argumentation mining aims to automatically identify structured argument data from unstructured natural language text. One particularly important aspect of argumentation mining is claim and evidence identification arising from abstracts of scientific articles, papers and documents, in order to extract useful information needed for better decision making and accurate planning controls. Therefore, in this project, our main goal is to exploit unstructured parsing information to detect claims and evidences coming from citations for biomedical literature from MEDLINE, life science journals and online books and which are referring in keywords related with the targets and indicators of the specific SDG. Three deep learning neural networks have been deployed as we mentioned in the previous section for the implementation of the specific approach and which have shown exceptional performances in various text classification tasks, including sentiment analysis, news categorization, topic classification, question answering (QA), and natural language inference (NLI), over the course of the past six years.

i) Previous efforts

There have been previous efforts to classify UN publications and facilitate document discovery and analytics. Specifically, DESA's Working Papers have recently been manually classified according to individual SDGs. There have also been a number of recent in-depth analyses of UN texts. Le Blanc, Freire, and Vierros (2017)[1], for example, use a large collection of UN publications and academic sources to manually determine the connections among the ten targets of SDG 14. Vladimirova and Le Blanc (2015)[2] used 40 global reports to carefully examine the links between education and other SDGs in flagship publications of the United Nations system. Le Blanc (2015)[3] analyzed the targets in each of the 17 SDGs that refer to multiple goals and show the connections between some thematic areas. In each of these novel papers, the authors demonstrated the power of expert analysis and careful reading of individual texts to derive important insights.

However, there are limits to how well this methodology can scale and how it can be replicated with other texts. For any significant number of texts, the time and focus needed to understand them all becomes prohibitive. The problem gets worse as the number of documents continues to grow and as one discovers new connections between topics. For example, a publication that discusses inequality touches upon unemployment, gender, social protection, vulnerability, public policy, and many other relevant topics. Moreover, major publications like DESA's World Economic and Social Survey cover a broad range of topics related to development and simultaneously address multiple SDGs. As the Latin and Greek aphorism tells us, art is long, life is short.

Furthermore, additional efforts have been made also from Sovrano, Palmirani and Vitali (2004)[4] as well as from Rodriguez Medina (2019)[5]. Specifically, as far as it concerns the first mentioned paper, they tried to produce a useful software for UN, that could help to speed up the process of qualifying the UN document following the SDGs in order to monitor the progresses at the world level to fight poverty, discrimination and climate changes using an automated labeling approach. As far as Rodriguez Medina (2019) is concerned, he tried to create and analyze a text classification dataset from freely-available web documents from the UN's SDGs and consequently used it to train and compare different multi-label text classifiers with the aim of exploring the alternatives for methods that facilitate the search of information of this type of documents.

---

[1] Le Blanc, David, Clovis Freire, and Marjo Vierros. 2017. "Mapping the Linkages between Oceans and Other Sustainable Development Goals: A Preliminary Exploration." DESA Working Paper 149 (February). https://www.un.org/development/desa/publications/working-paper/wp149

[2] Vladimirova, Katia, and David Le Blanc. 2015. "How Well Are the Links between Education and Other Sustainable development Goals Covered in UN Flagship Reports? A Contribution to the Study of the Science-Policy Interface on Education in the UN system." DESA Working Paper 146 (October). https://www.un.org/development/desa/publications/working-paper/education-and-sdgs-in-un-flagship-reports

[3] Le Blanc, David. 2015. "Towards Integration at Last? The Sustainable Development goals as a Network of Targets." DESA Working Paper 141 (March). https://www.un.org/development/desa/publications/working-paper/towards-integration-at-last

[4] Sovrano, Palminari, Vitali. 2004 "Deep Learning Based Multi-Label Text Classification of UNGA Resolutions". https://arxiv.org/ftp/arxiv/papers/2004/2004.03455.pdf

[5] Rodriguez Medina, Samuel. 2019. "Multi-Label Text Classification with Transfer Learning for Policy Documents: The Case of the Sustainable Development Goals". http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1360968&dswid=3657

ii) Approach of the problem

At this point, it seems logical as well necessary also, to present and elaborate further, our approach concerning the specific project. As we mentioned before, the main goal of this study, is to accurately predict the claims and evidences that displayed in the abstract part of documents, related with the targets and indicators of the SDG #3 of the UN in order to help governments and institutions to get better decisions and eliminate the time they need to read a paper. Therefore, we must first present very thoroughly and extensively the handling of this problem. First of all, in order to acquire all the necessary documents for our project, we implemented a web scraping approach by extracting the necessary data from the PubMed full-text archive of biomedical and life sciences journal literature of the U.S. National Institutes of Health's National Library. After that we used the certified online language processing service of "inception" that is part of the National Infrastructure for Language Resources and Technologies in Greece (CLARIN:EL), in order to properly annotate all the disposal documents that we had initially scrapped.

Subsequently, after all the necessary procedures required to format these documents in an appropriate type of data, we used as we mentioned before, three (3) deep neural network models. Neural network models have been demonstrated to be capable of achieving remarkable performance in sentence and document modeling. Convolutional neural network (CNN), Recurrent neural network (RNN) or Long Short-Term Memory Recurrent neural network (LSTM) and Feed Forward neural network (MLP) are three mainstream architectures for such modeling tasks, which adopt totally different ways of understanding natural languages. Specifically, a MLP neural network, is an artificial neural network wherein connections between the nodes do not form a cycle. As far as it concerns, a CNN model for text classification purposes is different from that of a neural network because it operates over a volume of inputs. Therefore, each layer tries to find a pattern or useful information of the data. Regarding also the RNN model is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior.

In this work we tried to set the best hyperparameters and learning rates to each model separately in order to acquire the best possible results and in order for each model to be trained as efficiently as possible. Furthermore, in order to define the structure of the models we used appropriate embedding and dense layers, as well as the necessary activation functions in order to get the corresponding activation values of each neuron and to normalize the outputs of the networks. It must be stressed that we followed a specific approach in all of our three models. Specifically, we implemented a stratified k fold validation in order to split the data in k datasets and in each iteration, we selected to keep outside one dataset a time, in order to train the models with the rest datasets. In order to evaluate the best model, we chose to keep the model with the least validation loss and subsequently with the greatest test accuracy. With this way, we managed to compare with a least biased method the capability of the models to cope with unknown data and therefore we did not rely on a random split and a model to train. Finally, in order to check the predictions of the best selected model, we used as a blind test case, a bunch of abstracts sentences derived from the PubMed as well and we tried to predict the label of each unique sentence.

3. Data

In this section of the project we will thoroughly present all the data we used for our analyses and for the deployment of our models, as well as all the pre-processing methods selected and the main statistical results derived after the necessary data transformation procedures.

i) Web scrapping

First of all, as we mentioned in the previous sub section of Chapter 2, we implemented a web scrapping approach, where by setting appropriately all the necessary queries for the selection of the right keywords that are referring to the SDG #3 we managed to acquire all the abstract texts from the PubMed full-text archive of biomedical and life sciences journal literature that were related with the targets and indicators of the specific SDG. Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites. Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser. Specifically, for the implementation of the specific procedure we used the Beautiful Soup library where is a Python library for pulling data out of HTML and XML files. Subsequently, we extract from all the disposal pages the url's and afterwards we extract and store each abstract and each title in a txt file. Finally, we save each unique filename, url and title of document in a csv file. This file was the file for all our metadata.

ii) Annotation procedure using the CLARIN:EL

After that procedure we ended up with a total 150 txt documents where with the help of the CLARIN:EL certified online language processing service of "inception", each member of our team, successfully annotated all these documents by giving each sentence the appropriate label that characterize them. Specifically, during this procedure we had to annotate properly the discourse topic of each abstract document, its argument i.e. the claim and evidence sentence or sentences as well as the research theme that characterize it. To be more specific, the topic sentence is what the discourse is about. A topic is usually found asking questions in general terms e.g., "What the text is actually about?", "What it reports?". We wanted actually to capture the aim, objective or the purpose of the paper. The topic ought to be indicative of the discourse and not to generic. Furthermore, we carefully noted to not annotate introductory expressions, the annotation span ideally should be a maximal nominal phase in a sentence, adjectives ought to be included in the beginning of an annotation span, possible punctuations at the end of an annotation span did not included, as well as we carefully noted that the topic and the title of each abstract to share common keywords.

Regarding the annotation of the argument we ought to consist a phrase with one or more claims that something is, or should be, the case and the evidence. The argument layer as we mentioned consisted from two labels. The claim and the evidence label. Generally, in a essay for example, argument consist of claims and premises. Premises can be supportive of attacking *a claim. In our discourse, and in the range of an abstract we don't face this distinctions. Because of the limited size, the authors state their major claim(s) and the evidence(s) that support it. Regarding also the claim label, we expected to find them in the end of the abstacts in the conclusion section, if the paper had sections, or in a sentence that concludes the results of the study. We should mentioned, that some abstracts did not include claim sentences and for that reason we exclude them from our analysis. As far as the evidence is referred to, it was the reason for accepting the conclusion. Generally, the evidence may include experiments, observations, experiences or ideas, and may not be work that the authors have conducted themselves. The evidence in our corpus was usually seen as a result, and in many cases a result from a statistical test. The difficulty here, was to distinguish evidences from results. The rules that we followed were the following. Many abstracts report numerical data and results from statistical test. Sentences with raw numerical data that inform about the population for example, are not evidences. On the other hand, the statistical results (significant or not) are very likely evidences that support a conclusion. Furthermore, claim and evidence sentences should share keywords, ideally also found in topic, and/or the title of the paper.*

Finally, regarding the research theme, we had to choose only one from a list of ten (10) disposal layers. Specifically, these were the following: Clinical Trial, Device, Diagnostic Tool, Drug, Infrastructure, Material, Prototype, Study, Treatment and Other. The Research Theme expresses the overarching goals of the authors and here we investigate how do they achieve it. Clinical trials are research studies performed in people that are aimed at evaluating a medical, surgical, or behavioral intervention. A medical device is any device intended to be used for medical purposes. Diagnosis is the identification of the nature and cause of a certain phenomenon. Diag-nosis is used in many different disciplines, with variations in the use of logic, analytics,and experience, to determine "cause and effect". nfrastructure is defined as "the basic facilities, services and installations needed forthe functioning of a community or society". A drug is any substance that causes a change in an organism's physiology or psychol-ogy when consumed. A biomaterial (or material in general) is any substance that has been engineered to inter-act with biological systems for a medical purpose - either a therapeutic (treat, augment,repair or replace a tissue function of the body) or a diagnostic one. A prototype is an early sample, model, or release of a product built to test a conceptor process. n an observational study, investigators assess health outcomes in groups of participantsaccording to a research plan or protocol. Finally, a therapy or medical treatment is the attempted remediation of a health problem, usu-ally following a diagnosis.

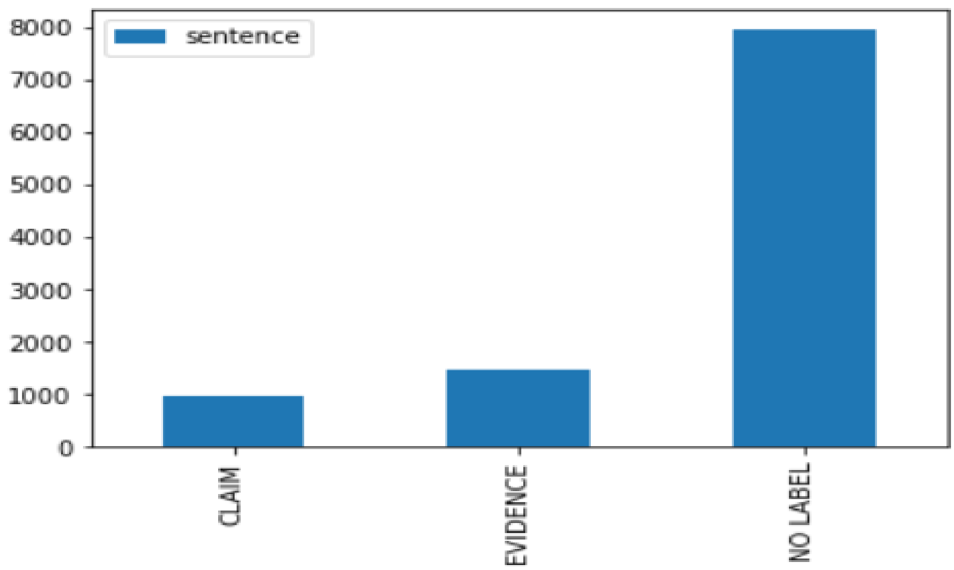
After the whole annotation procedure was done, we implemented a curation procedure in order to correctly annotate any documents that had no agreement according to the majority of annotators that were also annotated from the other part of a different team. This procedure was the final annotation procedure in order to conclude into documents with a powerful agreement. Therefore, after this procedure was finally ended, our final dataset was consisted from 900 csv files, where each file was a different abstract document and each row was consisted from the corresponding sentence of each document and from its characteristic label. More specifically, sentences that were lack of argument were characterized with "-" and each evidence and claim sentence were characterize with the corresponding label, i.e. "claim" and "evidence".

iii) Data Parsing

After having our final dataset in our hands, i.e. the 900 different csv files, with the corresponding format that was mentioned before, we had to parse them accordingly in order to be able to analyze them in a properly manner. For the specific purpose we build a corresponding parser code, where we set appropriately each document to be placed in a different row into a unique pandas data frame. Specifically, our data frame was consisted from two (2) columns. In the first column the label of the sentence was placed, while in the second column the sentence of each unique document was placed as well. Therefore, we ended up with 11064 rows and 2 variables. After having a quick view of our dataset we assumed that our data were very imbalanced, as a matter of fact 77,62% of our data were classified with no label and only 13,43% and 8,95% were classified as evidence and claim accordingly.

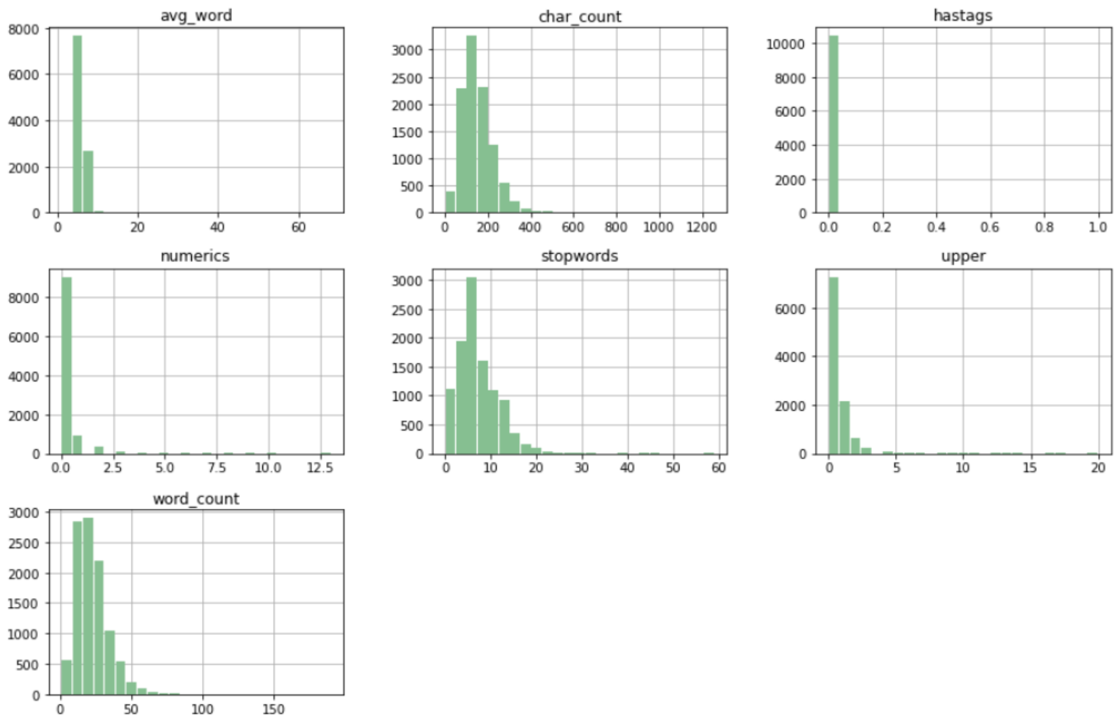
After that, we took special care so that our data did not consist of missing values and as we noticed they did not exist at all. Furthermore, as we observed there were many duplicate sentences, so we managed to get rid of them, as they would probably affect our final results. Therefore, we ended up with 10403 observations. As far as the distribution of our data is concerned after the final removal of the duplicated sentences, as we can observe from the following bar plot (Plot 1), we can assume that 7953 observations were classified as sentences with 'No Label', 1470 as 'Evidence' and 980 as 'Claim' sentences. In plain words, 635 sentences with the 'No label' layer were removed, 16 'evidence' sentences and 10 'claim' sentences were removed as well.

Plot 1: Distribution of the data



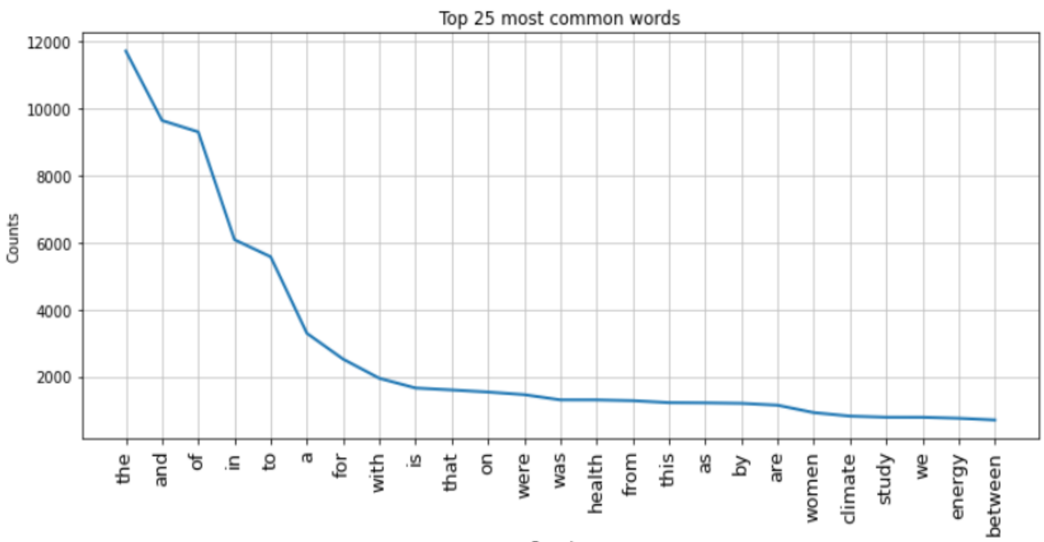
In order to have also a much more complete image of our dataset, we properly visualize using the following histograms (Plot 2), the distribution of the most common characteristics of our dataset, i.e. the number of words, characters, stop words, hastags, numbers and upper cases letters that appears in our sentences. As we can observe (Plot 2), the majority of our sentences were consisted from 10 - 30 words and on average from 6 – 8 words. Furthermore, the majority of our sentences as we can observe follow a right skewed distribution as the number of *characters and the number of stop words is referred to, with the majority of them to be consisted from 50 – 250 characters and 8 – 15 stop words. Finally, the number of hastags, upper cases letters and numbers inside the sentences were very few, from the matter of fact as we can the most of them were consisted under 5 unique corresponding abbreviations.*

Plot 2: Distribution of the most common characteristics of the sentences



Continuing our analysis, we found the most common words that appear in our sentences. As we can observe from the following line plot (Plot 3), the 25 most common words were the following, culminating in the word 'the', 'and', 'of', 'in' and 'to'.

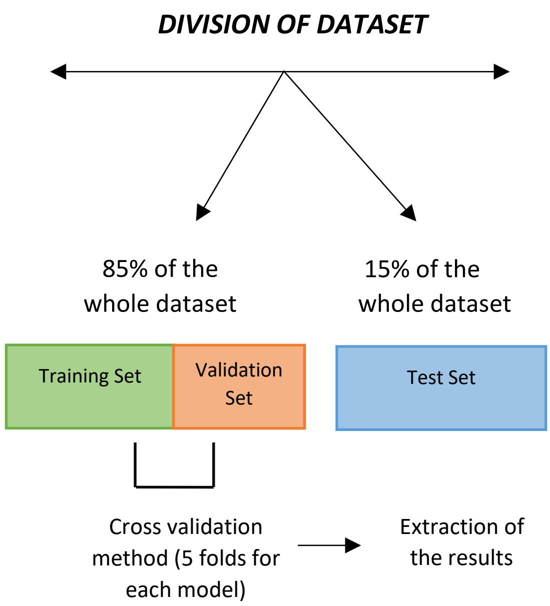
Plot 3: 25 most common words



iv) Pre-processing procedures

The first step that we had to implement before the deployment of our models, were any pre-processing procedures in the format of our data. Specifically, as a first step we proceed to the conversion of all upper cases letters to lower cases and we clean the text of the sentences from any unwanted symbols, such as from the removal of special characters and single characters and from the removal of any numbers that were inside the sentences. Furthermore, we remove all the unwanted stop words, as we see in a later stage they affected our results. After the implementation of the specific procedures, we had to declare properly the input variable (X) and the predicted output variable (y), where our input variable played the role of the sentences and the variable that we wanted to predict was obviously the corresponding label of our sentences.

Plot 4: Division of training, validation and test datasets



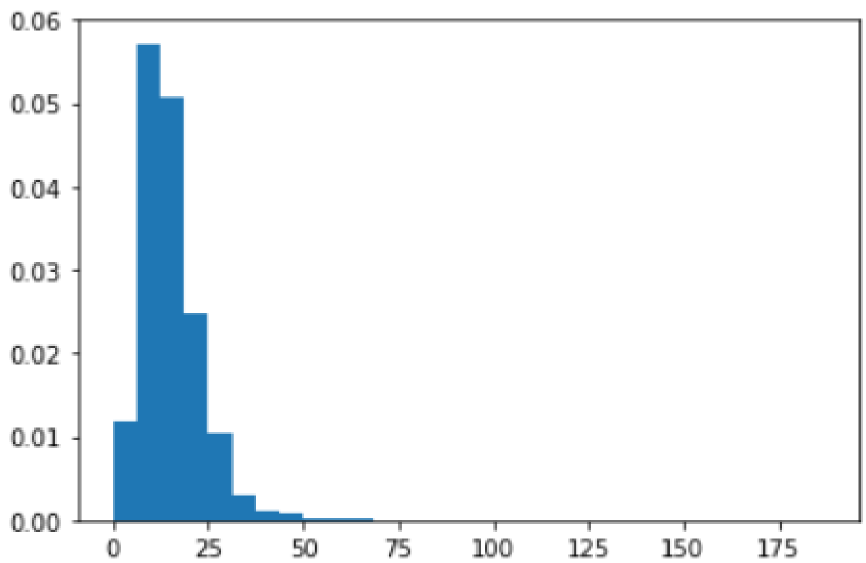
*For all the modelling procedures of our models a specific intervention logic was used. Specifically, before deploying our models, we tried to split our initial dataset into two parts. The first part was divided by **85%** of the initial dataset (**train/validation dataset**), which was used for the training and validation of our models, while the rest 15% was used for the predictive evaluation ability of the best model that was finally selected, which concerns data that had never been trained before. It should be mentioned also that the methodology we tried to follow for the experimental procedures of our models, was the stratified shuffle split and the cross validation method where we implemented 5 folds per iteration and the train – validations set were also split inside these folds. In statistics stratified shuffle sampling split is a sampling technique that is best used when a statistical population can easily be broken down into distinctive sub-groups. Furthermore, the data is shuffled every time and then are also split. In addition, cross-validation is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. Subsequently, in order to evaluate the best model, as we mentioned in a previous section of this report, we chose to keep the model with the least validation loss and subsequently with the greatest test accuracy. With this way, we managed to compare with a least biased method the capability of the models to cope with unknown data and therefore we did not rely on a random split and a model to train. Finally, in order to check the predictions of the best selected model, we used as a blind test case, a bunch of abstracts sentences derived from the PubMed as well and we tried to predict the label of each unique sentence.*

It must be stressed that during the splitting procedures, we fitted as well the tokenizer on the train dataset only and we proceed also to a padding procedure. At this point it seems appropriate to define what tokenization and padding is. To be more specific, tokenization is a common task in Natural Language Processing (NLP). It's a fundamental step in both traditional

NLP methods like Count Vectorizer and Advanced Deep Learning-based architectures. Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. These tokens are often loosely referred to as terms or words, but it is sometimes important to make a type/token distinction. A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing. A type is the class of all tokens containing the same character sequence. A term is a (perhaps normalized) type that is included in the IR system's dictionary. The set of index terms could be entirely distinct from the tokens, for instance, they could be semantic identifiers in a taxonomy, but in practice in modern IR systems they are strongly related to the tokens in the document. However, rather than being exactly the tokens that appear in the document, they are usually derived from them by various normalization processes.

Padding is a term relevant to convolutional neural networks as it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. For example, if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero. If, however, the zero padding is set to one, there will be a one pixel border added to the image with a pixel value of zero. Padding works by extending the area of which a convolutional neural network processes an image. The kernel is the neural networks filter which moves across the image, scanning each pixel and converting the data into a smaller, or sometimes larger, format. In order to assist the kernel with processing the image, padding is added to the frame of the image to allow for more space for the kernel to cover the image. Adding padding to an image processed by a CNN allows for more accurate analysis of images. As far as this procedure is referred to, we had first to find the maximum length for the 99% of all sentences. As we observed the 99% of the common length of the sentences were 43. This is also illustrated from the following histogram plot (Plot 5), where as we can observe the maximum length of the words is lower than 50 words. Therefore, for the implementation of our rest procedures we set as default the maximum length of the words to be 50 words long and as we observed the total number of words in the vocabulary was equal with 11988.

Plot 5: Distribution of the maximal length of the sentences



4. Methodology

In this project, three different models will be created to successfully identify the type of each sentence. Each model has its specific architecture and hyperparameters. It is noteworthy that the training data set is imbalanced. To train unbalanced classes 'fairly', we want to increase the importance of the two under-represented classes ("CLAIM", "EVIDENCE"). We will use a function from Keras, which automatically computes the class weights.

i) Model architecture and hyperparameters

A. First Model

The first model that was created, is a feedforward network. Feedforward Neural Network (FFNN) model is called a multilayer perceptron network that consists of input, hidden, and output layers of nonlinearly-activating nodes, which are interconnected in a feed-forward way with certain network weights. In FFNN, all the network weights are assigned random values initially, and the goal of the training is to adjust the set of network weights in a way that causes the output of the network to match the real values of the dataset as closely as possible.

To increase the performance of the model in terms of accuracy, an embedding layer was inserted into the architecture of the model so that it could learn the relations between unique words of the input data. It should be noted that while using this embedding method, each word is mapped to one vector instead of just an integer, and the vector values are learned jointly with the neural network model. So similar words in a semantic sense have a smaller (cosine) distance between them than words that have no semantic relationship. The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Backpropagation algorithm.

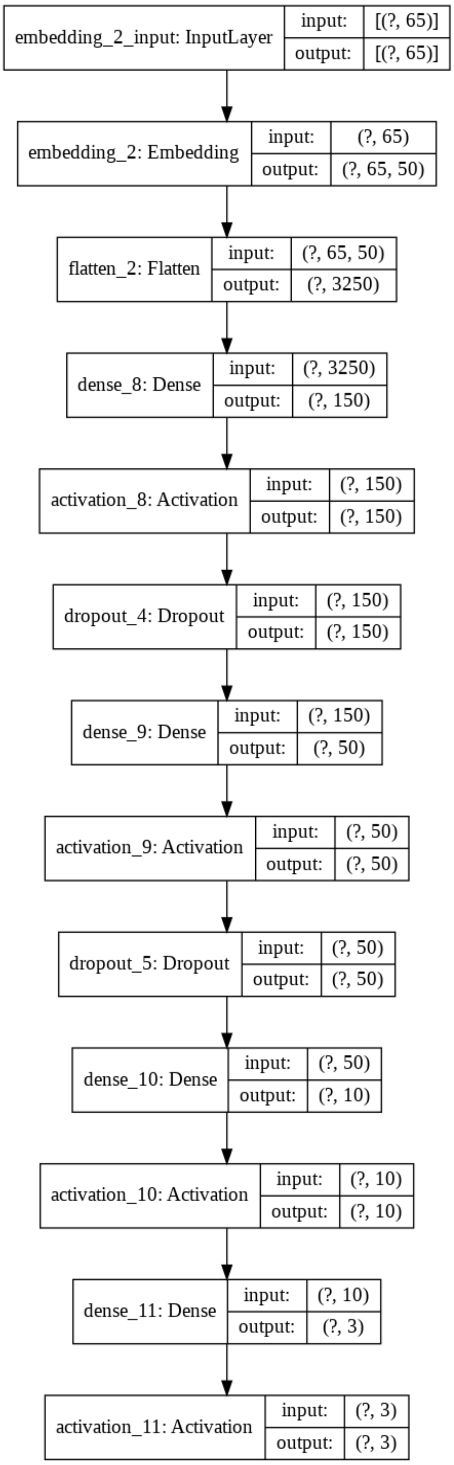
The hyperparameters that were changed to create the input and embedding layer of the model are the following:

Name of hyperparameter	Value	Description of hyperparameter
max_features	12270	This number indicates the size of the vocabulary
maxlen	65	The length of each sentence
embedding_dims	50	The size of each word embedding
batch_size	64	The number of training sentences in one forward/backward pass of the model
epoch number	40	The number of complete passes through the training dataset.

During each propagation, 64 vectorized sentences of length 65 are inserted into the model. The embedding layer changes the shape of the input data from 2D to 3D, meaning that the *shape of the samples will become 64x65x50 (50 represents the embedding dimensions of each word).*

➢ Model Architecture

Plot 6: Architecture of feed-forward neural network



The following plot (Plot 6) visualizes the architecture of the FFNN model that was trained to classify the data.

The specific model consists of an input layer that receives the vectorized data, whose length is equal to 65.

Then the embedding layer converts the 2D shape of the data to a 3D shape, as it transforms each word into a vector of size 50.

After the embedding layer, a Flatten layer is applied to flatten the input, meaning that the 3-dimensional input data will be transformed to 2D, without losing the information that was added by the embedding layer (the semantic relationship between the words).

Next, we add 3 dense layers and between them drop-out layers. The first dense layer consists of 150 neurons, the second layer consists of 50 neurons, and the third 10 neurons.

The 3 dropout layers that are located after each dense layer, have a drop-out rate of 30%. As a result, during training, some number of layer outputs are randomly ignored or “dropped out” *with a probability of 30%. This has the effect of making each dense layer be treated as a layer with a different number of nodes. This indicates that the neural network is forced to find new ways to classify the data, thus reducing the chance of overfitting. Without any dropout, our neural network exhibits substantial overfitting.*

Lastly, the output layer is added and is the one that outputs the classification predictions for each sentence. To be more specific it outputs a vector with a length of three since this project aims to classify the sentence into one of three classes. (“NO LABEL”, “CLAIM”, “EVIDENCE”). Each element of this vector is a probability, which shows the possibility of a sentence to belong to a specific class.

It should also be noted that for each dense layer, an activation layer was applied with the Relu function. Since the problem of this project is a multiclass classification problem, we use the Softmax function to output the predictions.

- Number of parameters

The total number of parameters, which are tuned during the learning procedure of the model, is also calculated in the following paragraph.

Since each distinct word is represented as a 50-sized vector, the total number of parameters that are created by the embedding layer is equal to :

$$vocabulary\ size\ \times embedding\ dimensions = 12270\ \times 50 = 613500$$

The input data that the Flatten layer receives has shape (none, 65, 50) and then it is converted to (none, 65x50 = 3250). Each value of the resulting input data is connected with each neuron of the 1st hidden layer via a specific weight. So the number of parameters that are going to be created in the 1st layer is:

$$input\ dimensions\ \times number\ of\ neurons\ of\ 1st\ layer = 3250\ \times 150 = 487500$$

It is important to note that in each multiplication between the input and the weights of each neuron, another parameter is added to the equation, which is named "bias parameter". So the total number of parameters in the 1st layer is the following :

$$\begin{gathered} input\ dimensions\ \times number\ of\ neurons\ of\ 1st\ layer + bias\ parameters \\ = 3250\ \times 150 + 1\ \times 150 = 487650 \end{gathered}$$

The input data that the 2nd hidden layer receives has shape (none, 300), where 300 are the dimensions of the 1st layer's output. So the number of new parameters that have been added is :

$$\begin{gathered} input\ dimensions\ \times number\ of\ neurons\ of\ 2nd\ layer + bias\ parameters \\ = 150\ \times 50 + 1\ \times 50 = 7550 \end{gathered}$$

Furthermore, new parameters have been added from the 3rd layer and their count is :

$$\begin{gathered} input\ dimensions\ \times number\ of\ neurons\ of\ 3rd\ layer + bias\ parameters \\ = 50\ \times 10 + 1\ \times 10 = 510 \end{gathered}$$

Lastly, the total number of parameters increases by 33, due to the multiplication of the output dimensions of the last hidden layer (none, 10) with the 3 neurons of the output layer. In total, the model contains 1109243 trainable parameters.

- Optimizer and Evaluation Metrics

For the optimizer, we used the Adam optimization algorithm. Since this model is sensitive to overfitting, we also decreased the learning rate to 0,005%. Note that the learning rate controls how quickly or slowly the neural network model learns the data. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train. For evaluation metrics we made use of the Categorical crossentropy loss function and categorical accuracy.

B. SECOND MODEL

The second model that was created, is a convolutional network with an embedding layer. In a traditional feedforward neural network, each input neuron is connected to each output neuron in the next layer. In contrast, the CNN model uses convolutions over the input layer to compute the output. This convolutional network contains one 1D convolutional layer. This convolutional layer contains a series of filters known as convolutional kernels. Each convolutional kernel is like a sliding window, whose job is to look at embeddings for multiple words. Each kernel is designed to look at a word, and surrounding words in a sequential window, and output a value that captures something about that phrase. To set up a network so that it is capable of learning a variety of different relationships between words, it is crucial to provide the model with many filters.

The hyperparameters that were adjusted to create the input, embedding layer, and the 1D convolutional layer of the model are the following:

Name of hyperparameter	Value	Description of hyperparameter
max_features	12270	This number indicates the size of the vocabulary
maxlen	65	The length of each sentence
embedding_dims	50	The size of each word embedding
nof_filters	64	The number of kernels
kernel size	3	The height of each kernel
batch_size	64	The number of training sentences in one forward/backward pass of the model
epoch number	60	The number of complete passes through the training dataset.

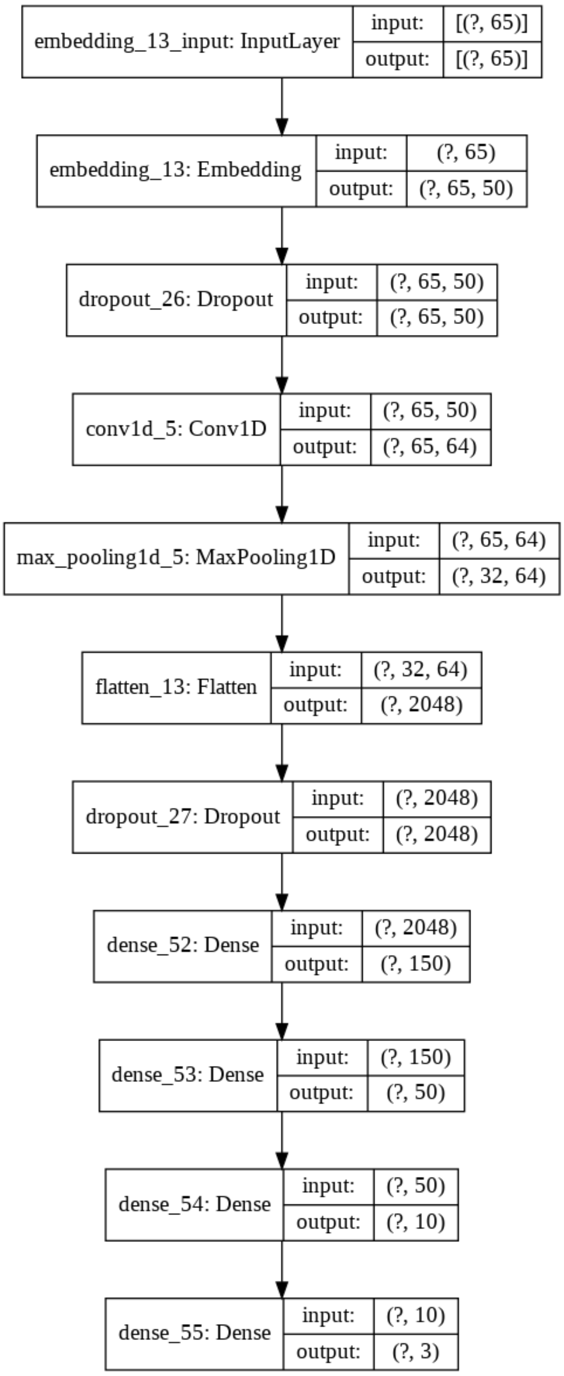
Note that the kernels will no longer be square, instead, they will be a wide rectangle with dimensions kernel_size x embedding_dims = 3 x 50.

The height of the kernel will be the number of embeddings it will see at once, similar to representing an n-gram in a word model. In this case, it will look at each 3-gram in each sentence.

- <u>Model Architecture</u>

The following plot (Plot 7) visualizes the architecture of the CNN model that was trained to classify the data.

Plot 7: Architecture of Convolutional neural network



The specific model consists of an input layer that receives the vectorized data, whose length is equal to 65.

Then the embedding layer converts the 2D shape of the data to a 3D shape, as it transforms each word into a vector of size 50.

After the creation of the embedding layer, we set a Dropout layer with a dropout rate = 40%.

The output (sentence) of the embedding layer is shaped as a 65x50 matrix. This matrix acts as the input layer for the 1D convolutional layer. The elements involved in carrying out the convolution operation are the kernels/filters. Every filter performs convolution on the sentence matrix and generates (variable-length) feature maps. It is worth mentioning that the output's height of the convolutional layer has not changed as it would be, if we didn't use padding. Since each kernel outputs a specific vector after the convolution, the width of the final output will be equal to the number of filters (64).

If we didn't use padding, the output's shape of the convolutional layer would be (63, 64), meaning that the height of the final output would be 63 instead of 65. This happens because each filter is applied systematically to the input sentence matrix. It starts from the top of the matrix and is moved from top to bottom by one row at a time until the edge of the filter reaches the bottom edge of the sentence matrix. For a 3x50 filter applied to a 65x50 sentence matrix, we can see that it can only be applied 63 times. The problem with this *phenomenon is that the kernels didn't pay attention to the edges of the input matrix and disregarded that information (first and last word). The reduction in the size of the output to the feature map is referred to as border effects (1).*

After the convolutional layer was applied, a 1-max pooling is performed over each map and the largest number from each feature map is recorded. According to the, the MaxPooling1D layer (2) "downsamples the input representation by taking the maximum value over the window defined by pool_size". In our case, the pool size is equal to two. So the height of the output would be equal to the height of the input divided by two.

A Flatten layer is then used to flatten the input, resulting to an output that has a shape of (none, 32 x 64) = (none, 2048). A dropout layer is also applied with the same dropout rate as the previous one.

The last four layers that we use are dense layers. The first layer contains 150 neurons, the second 50 neurons, and the third one 10 neurons. The last layer is the output layer which in turn outputs the classification predictions for each sentence.

- <u>Number of parameters</u>

The total number of parameters, which are tuned during the learning procedure of the model, is also calculated in the following paragraph.

Since each distinct word is represented as a 50-sized vector, the total number of parameters that are created by the embedding layer is equal to :

$$vocabulary\ size\ \times embedding\ dimensions = 12270\ \times 50 = 613500$$

To calculate the learnable parameters in the 1D convolutional layer, we have to multiply by the shape of width, height of current filters and number of previous layer's filters. There is also a bias term for each of the filter. In this case, there are no previous filters so the variable previous layer's filters is zero.

*Number of parameters in this CONV layer would be : ((shape of width of the filter * shape of height of the filter +1)*number of filters)*

*((shape of width of the filter * shape of height of the filter +1)*number of filters) = ((3 × 50 + 1) × 64 = 9664)*

After the Max-Pooling layer the shape of the input will be (32, 64). The Flatten layer then is applied an d the input's dimensions are equal to 32 × 64 = 2048.

Lastly, 3 dense layers and the output layer are applied, which in turn add $307350 + 7550 + 510 + 33 = 315443$ *new trainable parameters. . In total, the model contains 938607 trainable parameters.*

- <u>Optimizer and Evaluation Metrics</u>

For the optimizer, we used the Adam optimization algorithm. Since this model is sensitive to overfitting, we also decreased the learning rate to 0,005%. For evaluation metrics we made use of the Categorical crossentropy loss function and categorical accuracy.

C. THIRD MODEL

The third model that was created, is a Bidirectional LSTM model. A Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture, which according to Wikipedia (3) manages to "deal with the vanishing gradient problem that can be encountered when training traditional RNNs". Traditional RNNs suffer from the problem of vanishing gradients,which hampers learning of long data sequences. The advantage of an LSTM cell compared to a common recurrent unit is its cell memory unit. The cell vector has the ability to encapsulate the notion of forgetting part of its previously stored memory, as well as to add part of the new information. Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.

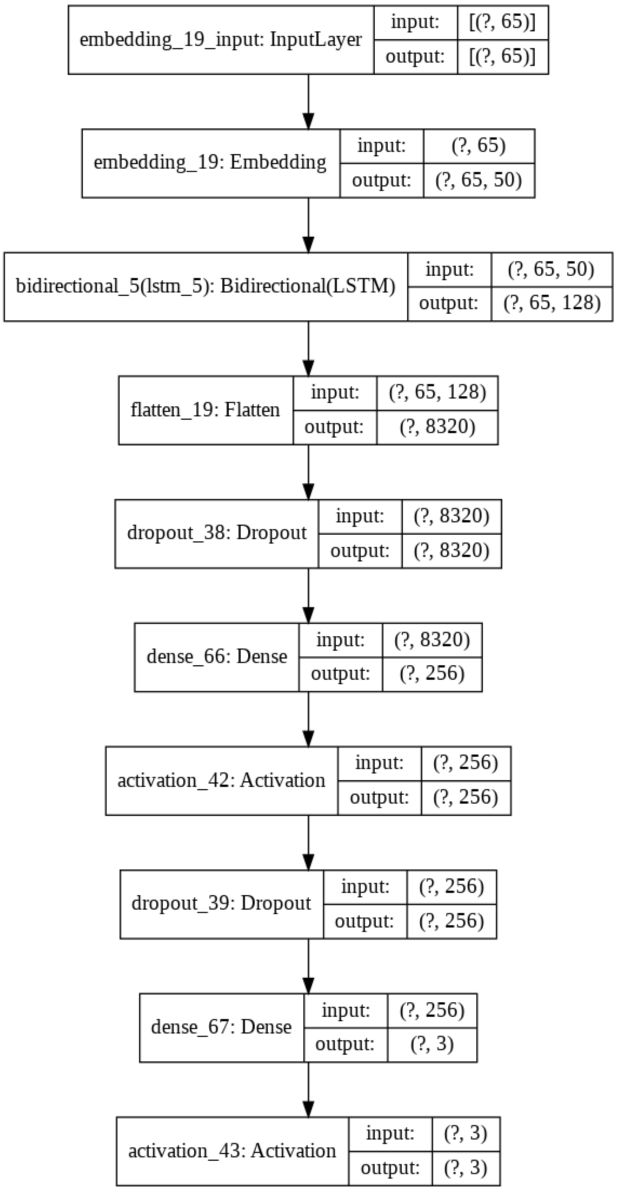
The hyperparameters that were adjusted to create the architecture the RNN model are the following:

Name of hyperparameter	Value	Description of hyperparameter
max_features	12270	This number indicates the size of the vocabulary
maxlen	60	The length of each sentence
embedding_dims	50	The size of each word embedding
batch_size	32	The number of training sentences in one forward/backward pass of the model
epoch number	10	The number of complete passes through the training dataset.
memory units	64	The number of memory units for each LSTM

It should be mentioned that one bidirectional LSTM layer wiil be created, meaning that two instead of one LSTMs will be trained on the input sequence and each one will contain 64 memory units (in total 128 units).

- Model Architecture

Plot 8: Architecture of Bidirectional LSTM model



The specific model consists of an input layer that receives the vectorized data, whose length is equal to 65.

Then the embedding layer converts the 2D shape of the data to a 3D shape, as it transforms each word into a vector of size 50.

In Bi-LSTM there will be one LSTM unrolling from left to right (LSTM1) on the input sentence matrix and another LSTM unrolling form right to left (LSTM2). Since the input size is (n,65,50), where n refers to the batch size, LSTM1 will return output of size (n,65,64)and LSTM2 will also return output of size (n,65,64).

The layer will then combine the output of LSMT1 and LSTM2 by applying element wise concatenation of LSTM1 output to LSTM2 at each timestep. This will result in an output of size (n,65,128).

A Flatten layer is then used to flatten the input, resulting to an output that has a shape of (none, 65 x 128) = (none, 8320).

After the creation of the Flatten layer, we set a Dropout layer with a dropout rate = 50%.

A dense layer is also added to the model and it contains 256 neurons. A dropout layer is also applied with the same dropout rate as the previous one. The last layer is the output layer, which outputs the classification predictions for each sentence.

➢ Number of parameters

The total number of trainable parameters that are created by the embedding layer is equal to :

$$vocabulary\ size\ \times embedding\ dimensions = 12270\ \times 50 = 613500$$

The number of trainable parameters that are created by the bidirectional LSTM layer are calculated by the following equation :

$$4\ \times \left((size\ of\ input + bias\ term) \times size\ of\ output + size\ of\ output^2\right)\ \times 2$$

The ×2 is added because the bidirectional LSTM contains two LSTMs.

So the number of new trainable parameters is equal to :

$$4 \times \left((50+1) \times 64 + 64^2\right) \times 2 = 58880$$

The Flatten layer that is applied later, returns a vector output of length 6400 for each sentence. The dense layer which is located after the Flatten layer has 256 neurons. This means that the number of new trainable parameters from the Dense layer will be equal to :

$$8320 \times 256 + 1 \times 256 = 2130176$$

.

The last new trainable parameters will be created in the output layer and will number :

$$256 \times 3 + 1 \times 3 = 771$$

In total, the model contains 2803327 trainable parameters.

- <u>Optimizer and Evaluation Metrics</u>

For the optimizer, we used the Adam optimization algorithm. Since this model is sensitive to overfitting, we also decreased the learning rate to 0,005%. Note that the learning rate controls how quickly or slowly the neural network model learns the data. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train. For evaluation metrics we made use of the Categorical crossentropy loss function and categorical accuracy.

ii) Model comparison

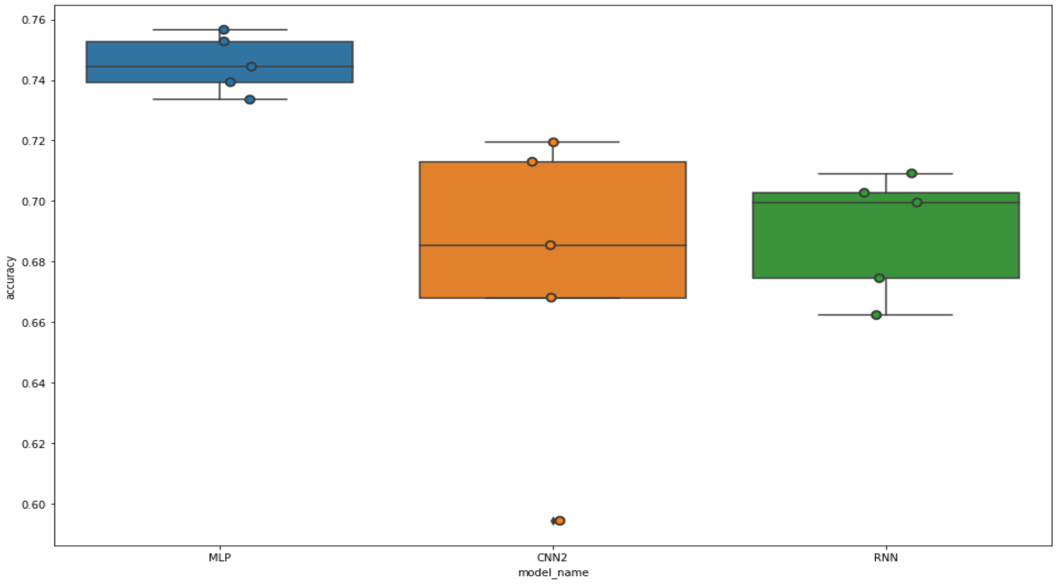
The dataset has been splitted to two sets. The train-validation dataset and the test dataset.

The ModelCheckpoint callback function (4) is used in conjunction with the training of each model to save a model or weights (in a checkpoint file) at some interval, so the model (or weights) can be loaded later to test its performance. This function stores the best model where the validation loss is least among all epochs.

We will use the test dataset, to test the performance of each model that was created. It is important to note that we will mostly concentrate on the accuracy of the minority classes to identify the best model.

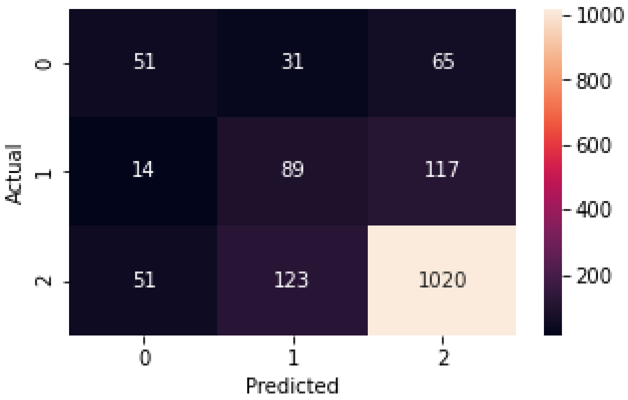
5. Results

After training our models with 5-fold cross validation and by keeping the accuracy of each training, we end up with the following box plot containing the results of each model in the test data-set.



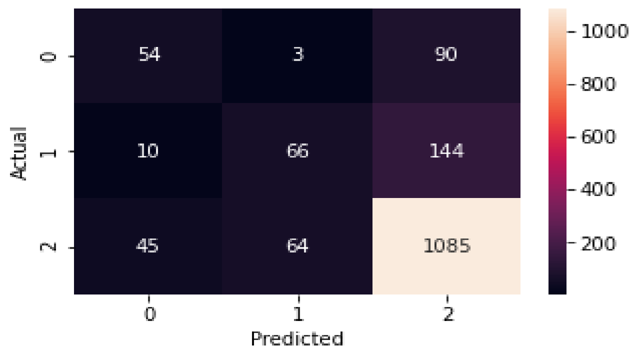
The plot above indicates that RNN and CNN are not the appropriate models to use and MLP has better results. Accuracy is not the best metric for selecting a model that makes good predictions for each label. In order to further analyze the competence of each model, the following confusion matrices and tables were made.

MLP confusion matrix and table



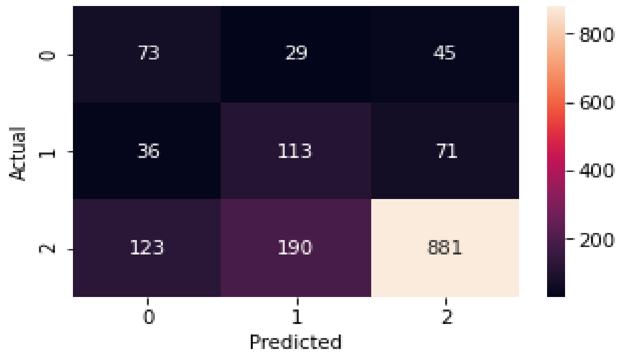
	precision	recall
CLAIM	0.43	0.41
EVIDENCE	0.40	0.52
NO LABEL	0.87	0.82
accuracy		
macro avg	0.57	0.59
weighted avg	0.76	0.74

CNN confusion matrix and table



	precision	recall
CLAIM	0.31	0.54
EVIDENCE	0.40	0.38
NO LABEL	0.87	0.80
accuracy		
macro avg	0.53	0.57
weighted avg	0.75	0.72

RNN confusion matrix and table



	precision	recall
CLAIM	0.31	0.50
EVIDENCE	0.34	0.51
NO LABEL	0.88	0.74
accuracy		
macro avg	0.51	0.58
weighted avg	0.75	0.68

Two metrics are displayed in the above tables' precision and recall. The first one is the percentage of the right predictions classified by each model for every label.

$$\frac{TP}{\mathrm{TP} + \mathrm{FP}}$$

The other one is the percentage of the actual labels being predicted correctly (the sensitivity of our model in finding the evidence and claims). Both are important metrics because we want to chose a model that makes good predictions for each label.

$$\frac{TP}{\mathrm{TP} + \mathrm{FN}}$$

Where

TP = the number of true positives for a specific label

FP = the number of false positives for a specific label

FN = the number of false negatives for a specific label

MLP has 43% precision for claims and 40% for evidence, results better than those of CNN (with 31% for claims and 40% for evidence) and RNN (with 31% for claims and 34% for evidence). With respect to recall RNN has astoundingly better results with 50% and 51%. In this case if we wanted to have a model that misses less evidence and claims, but makes more misclassifications in the 'no label' then the RNN would be our choice and this is the reason that accuracy is not always the best metric to use. In our case precision is more important and for this reason MLP is the final model we chose, which has overall better precision and recall.

Appendix

i) Members/Roles

- <u>ΑΓΑΠΙΟΥ ΜΑΡΙΟΣ</u>

  1. Web Scrapping Script on PubMed site & Data collection
  2. Annotation & Curation
  3. Data Parsing & Preprocessing & Model Construction & Model Evaluation
  4. Report the Methodologies used to create the models

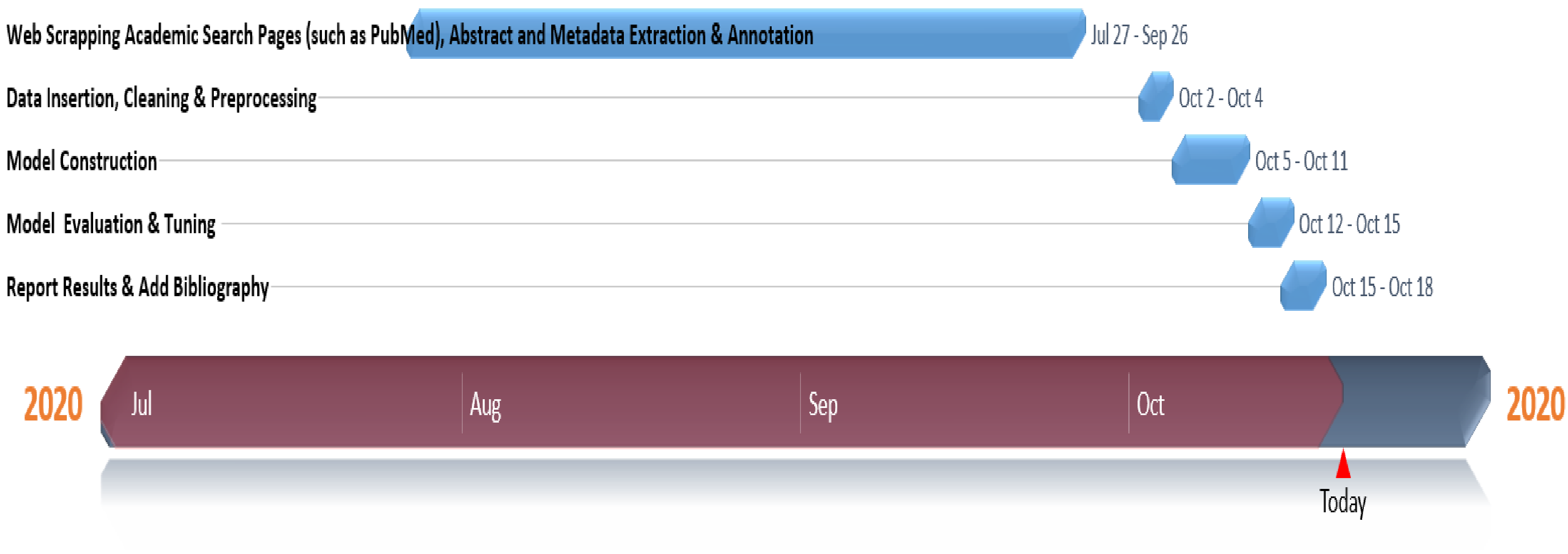
- <u>ΚΟΝΤΟΣ ΧΡΗΣΤΟΣ</u>

  1. Data collection
  2. Annotation & Curation
  3. Data Parsing & Descriptive Statistics & Preprocessing
  4. Report the Business Problem Description, the Mission and Data Preprocessing

- <u>ΣΚΑΡΛΗΣ ΒΑΣΙΛΗΣ</u>

  1. Data collection
  2. Annotation
  3. Data Parsing & Preprocessing & Model Construction & Model Evaluation
  4. Report the Evaluation & Comparison of the models

ii) Time Plan


Web Scrapping Academic Search Pages (such as PubMed), Abstract and Metadata Extraction & Annotation
Jul 27 - Sep 26
Data Insertion, Cleaning & Preprocessing
Oct 2 - Oct 4
Model Construction
Oct 5 - Oct 11
Model Evaluation & Tuning
Oct 12 - Oct 15
Report Results & Add Bibliography
Oct 15 - Oct 18
2020
Jul
Aug
Sep
Oct
2020
Today

Bibliography

1. *Jason Brownlee. A Gentle Introduction to Padding and Stride for Convolutional Neural Networks. machinelearningmastery.com. [Online] April 19, 2019 . https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/.*
2. ***Jason Brownlee.** A Gentle Introduction to Padding and Stride for Convolutional Neural Networks. machinelearningmastery.com. [Online] April 19, 2019 . https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/.*
3. *Long short-term memory. Wikipedia. [Online] https://en.wikipedia.org/wiki/Long_short-term_memory.*
4. *ModelCheckpoint. Keras. [Online] https://keras.io/api/callbacks/model_checkpoint/.*
5. *https://arxiv.org/pdf/1706.00188v1.pdf*
6. *https://paperswithcode.com/paper/a-c-lstm-neural-network-for-text*
7. *https://www.ijitee.org/wp-content/uploads/papers/v8i6s/F60970486S19.pdf*
8. *http://www.ijaerd.com/papers/finished_papers/Sentiment%20Analysis%20with%20Multilayer%20Perceptron%20using%20Emoticon%20Space%20Model-IJAERDV04I0760860P.pdf*
9. *https://www.researchgate.net/publication/336058961_Affective_Computing_and_Deep_Learning_to_Perform_Sentiment_Analysis*
10. *https://www.researchgate.net/publication/327682642_Investigation_of_Naive_Bayes_combined_with_Multilayer_Perceptron_for_Arabic_sentiment_analysis_and_opinion_mining*
11. *https://arxiv.org/abs/1408.5882*
12. *https://paperswithcode.com/paper/convolutional-neural-networks-for-sentence*
13. *https://ai.stanford.edu/~ang/papers/ICPR12-TextRecognitionConvNeuralNets.pdf*
14. *https://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf*
15. *https://paperswithcode.com/paper/an-end-to-end-trainable-neural-network-for*
16. *https://paperswithcode.com/paper/multi-branch-attentive-transformer*
17. *https://paperswithcode.com/paper/sentence-state-lstm-for-text-representation*
18. *https://paperswithcode.com/paper/revisiting-lstm-networks-for-semi-supervised-1*
19. *https://arxiv.org/pdf/1901.06610v2.pdf*
20. *http://jens-lehmann.org/files/2019/epia_simple_lstm.pdf*
21. *https://arxiv.org/pdf/1810.03660v1.pdf*
22. *https://en.wikipedia.org/wiki/Feedforward_neural_network*
23. *https://en.wikipedia.org/wiki/Convolutional_neural_network*
24. *https://en.wikipedia.org/wiki/Recurrent_neural_network*
25. *https://www.researchgate.net/publication/43121576_A_Review_of_Machine_Learning_Algorithms_for_Text-Documents_Classification*
26. *https://www.researchgate.net/publication/228084521_Text_Classification_Using_Machine_Learning_Techniques*
27. *https://repositorio-aberto.up.pt/bitstream/10216/107762/2/219435.pdf*

28. *https://medium.com/@jusky8/text-classification-using-machine-learning-5838293c9f2*
29. *https://arxiv.org/pdf/2004.03705.pdf*
30. *https://arxiv.org/pdf/1810.03660v1.pdf*
31. *https://zindi.africa/competitions/sustainable-development-goals-sdgs-text-classification-challenge*
32. *https://www.un.org/esa/desa/papers/2019/wp159_2019.pdf*
33. *https://www.scitepress.org/Papers/2018/67305/67305.pdf*
34. *https://academic.oup.com/database/article/doi/10.1093/database/baz034/5424138*
35. *https://unctad.org/system/files/non-official-document/PPT%20for%20UNCTAD%20Stats%20-%20by%20Richard%20Rothenberg%20-Feb03-2020.pdf*
36. *https://alexmoltzau.medium.com/undp-the-sdgs-and-the-role-of-innovation-involving-machine-learning-ae4ce920d592*
37. *https://www.ijcai.org/Proceedings/15/Papers/033.pdf*
38. *https://inception-project.github.io//*
39. *https://www.clarin.gr/en/about/what-is-clarin*
40. *https://sdgs.un.org/goals*
41. *Le Blanc, David, Clovis Freire, and Marjo Vierros. 2017. "Mapping the Linkages between Oceans and Other Sustainable Development Goals: A Preliminary Exploration." DESA Working Paper 149 (February). https://www.un.org/development/desa/publications/working-paper/wp149*
42. *Vladimirova, Katia, and David Le Blanc. 2015. "How Well Are the Links between Education and Other Sustainable development Goals Covered in UN Flagship Reports? A Contribution to the Study of the Science-Policy Interface on Education in the UN system." DESA Working Paper 146 (October). https://www.un.org/development/desa/publications/working-paper/education-and-sdgs-in-un-flagship-reports*
43. *Le Blanc, David. 2015. "Towards Integration at Last? The Sustainable Development goals as a Network of Targets." DESA Working Paper 141 (March). https://www.un.org/development/desa/publications/working-paper/towards-integration-at-last*
44. *Sovrano, Palminari, Vitali. 2004 "Deep Learning Based Multi-Label Text Classification of UNGA Resolutions". https://arxiv.org/ftp/arxiv/papers/2004/2004.03455.pdf*
45. *Rodriguez Medina, Samuel. 2019. "Multi-Label Text Classification with Transfer Learning for Policy Documents: The Case of the Sustainable Development Goals". http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1360968&dswid=3657*