MACHINE LEARNING AND CONTENT ANALYTICS

Spring Quarter 2020-2021 (Part-Time)

Argument Mining Project

Kasma Maria Anna (f2822023)
Tsekouras Ilias (f2821914)
Tsiolis Spyridon (f2821915)

Contents

1 Introduction

"Words are the new numbers". This quote (Thorsrud 2018) highlights the power of the written message in the academic discourse. Scientific knowledge is produced at an accelerated pace making its discovery and assessment a challenging task. Natural language processing (NLP) technologies, in general, and text-mining tools, in particular, have become increasingly useful in order to identify and characterize the most relevant information produced in a given scientific discipline. The logic, rhetoric and dialectic quality dimensions of a scientific paper are important parts to be considered in order to assess them Hence, not only the claims made by its authors, but also the evidence that they provide as a supporting material have to be identified.

Argumentation mining aims to automatically identify structured argument data from unstructured natural language text. This challenging, multifaceted task is recently gaining a growing attention, especially due to its many potential applications. One particularly important aspect of argumentation mining is claim identification.

The (Context-Independent) Claim Detection problem (CD) is an information extraction task, whose goal is to identify those sentences in a document that contain the conclusive part of an argument. According to Walton [2009], an argument is a set of statements which consists in three parts: a conclusion, a set of premises, and an inference from the premises to the conclusion. While this definition is widely accepted, these concepts have also been defined in the literature with different names: conclusions could be referred to as claims, premises are often called evidence or reasons, and the link between the two, i.e., the inference, is sometimes called the argument itself. CD is a quite subtle task, even for humans, since the concept itself of claim could hardly be framed by a compact set of well-defined rules. Indeed, in some cases claims consist of opinions supported by some evidence facts, while in other cases they are represented by statements that describe relevant concepts [Levy et al., 2014], as it happens, e.g., in legal domains.

2 Our project

Based on abstracts of published scientific papers, the aim of this project is the prediction of the annotated labels, that is the claim, the evidence or neither (No label) of their sentences, as not all the abstracts contain claims and evidences in their content.

The publication of a scientific paper does not imply the generation of new information or powerful insights, which in turn does not provide any new claim supported by the respective evidence. Hence, the goal of the algorithm produced in this assignment is to verify the power of scientific paper only through its abstract, serving as a useful tool for the decision making of

a government's strategy plan in various domains and for the continuation of scientific researches in universities.

3 Methodology

3.1 Text classification

To face the text classification problem we introduced a baseline model as well as two neural network models so that we can compare them. Regarding the baseline model, we produced a word lexicon based on key words in order to identify claim or evidence sentences. Also, we applied a rule that considered the last sentence of a given abstract as a claim sentence. As for the two Convolutional neural networks that we decided to use, the one concerned a pre-trained model, called GloVe-300 and the other was trained by our team.

3.2 Document clustering

For the Document Clustering Problem, 2 models were utilized. The GloVe-300 model was again utilized. The second model was trained by our team, using the Word2Vec model.

4 Input Data Processing

4.1 Data Source

The data utilized for the project's purposes comprised of abstracts of published scientific papers gathered from Horizon 2020, the financial instrument implementing the Innovation Union, a Europe 2020 flagship initiative aimed at securing Europe's global competitiveness. The abstracts originated from different scientific fields, concerning environmental, biomedical and other scientific researches, in order to boost the algorithm's accuracy.

4.2 Data preparation

4.2.1 Data Annotation

After the abstracts were extracted by the coordinator of the project, Mr. Aris Fergadis, they were distributed to all teams which chose the pilot project in order to proceed with the annotation phase. Hence, the task was to identify and annotate the argumentative statements, that is the claim and the evidence, for each abstract. It should be noted that the annotation span in our task was always a sentence. Reading only the claim and the evidence from an abstract should make sense and those snippets should look like a summary of the abstract.

Scientific publications have a structure that most authors follow. This structure have differences depending on the scientific domain. However, the generic sections are the following:

1. Background
2. Objective/Aim
3. Methods
4. Results
5. Conclusions

The Label Studio, an open source data labeling tool for labeling and exploring multiple types of data, was utilized for the annotation phase. For each abstract, each team member had to identify the structure. Each sentence was given one of the above five categories. Then, the claim and the evidence were identified. Afterwards, the citances, which are the sentences that refer to the work of other authors, were annotated based on the following four categories:

1. Positive – Supporting (showing that the author's base / extend /verify the claim or the findings / results of the abstract),
2. Negative – Contrasting (stating that the authors question / argue / dispute / controvert the claim or the findings of the abstract).
3. Neutral – Mentioning (mentioning the reference paper without a standpoint)
4. Irrelevant (The citation seems to have nothing in common with the reference article)

, in order to identify the standpoint of the author to the cited work.

Then, the reliability of each team's annotations was assessed by considering inter-annotators' agreement. The MACE classifier was used, which learns competence estimates for each annotator and computes the most likely answer based on them.

4.2.2 Dataset Overview

Having completed the annotation phase, we concluded to 1,017 abstracts and a total of 9,558 sentences. Then, each abstract along with its labelled sentences were imported and all of them were concatenated to produce the final dataset, which comprised of 2,686 abstracts, 32,004 sentences (rows) and three columns: the document id of the abstract, the sentence and the label.

Regarding the latter, the value of 0 corresponds to the "No label" argumentative statement, the value of 1 to the "Evidence" argumentative statement and the value of 2 to the "Claim" argumentative statement. A preview of the top 5 rows of the final dataset is presented in Figure 1.

	doc_id	sentence	label
0	0	Concordance Between Different Amyloid Immunoassays and Visual Amyloid Positron Emission Tomographic Assessment	0
1	0	Importance Visual assessment of amyloid positron emission tomographic (PET) images has been approved by regulatory authorities for clinical use.	0
2	0	Several immunoassays have been developed to measure β-amyloid (Aβ) 42 in cerebrospinal fluid (CSF).	0
3	0	The agreement between CSF Aβ42 measures from different immunoassays and visual PET readings may influence the use of CSF biomarkers and/or amyloid PET assessment in clinical practice and trials.	0
4	0	Objective To determine the concordance between CSF Aβ42 levels measured using 5 different immunoassays and visual amyloid PET analysis.	0

Figure 1: Top-5 rows of the final dataset

Out of the 32,004 sentences, 22,375 sentences corresponding to the 69.9% of the final dataset had no label, 6,210 sentences (19.4%) were annotated as evidence and 3,419 sentences (10.7%) were annotated as claim, as it is depicted in Figure 2.

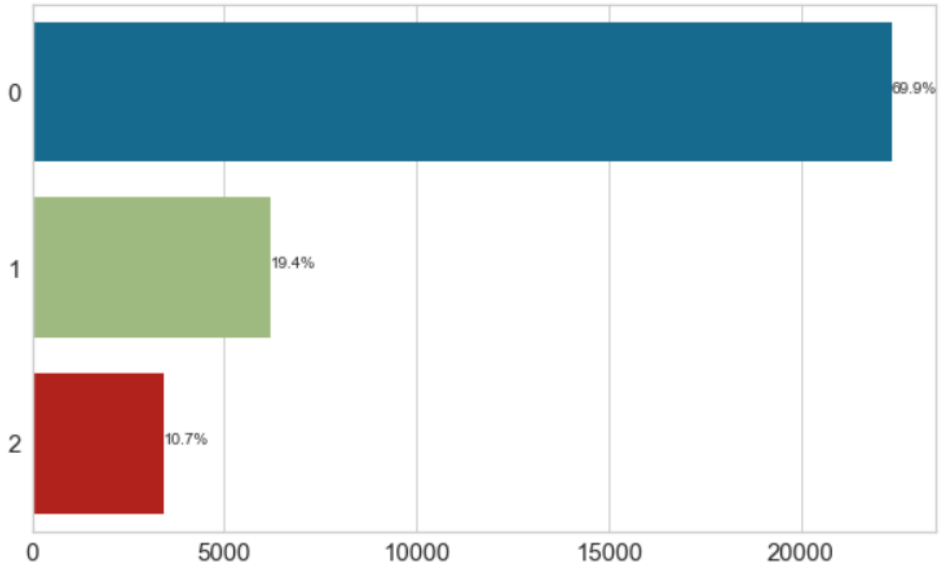

Figure 2: Labels' frequency in the final dataset

5 Baseline Model

Prior to implementing any machine learning model, as a first approach, we created a baseline model that was based on rules and patterns that we had identified during our annotation task, in order to identify claim or evidence sentences.

As first rule we created two separate lexicons which contained keywords for each of the two sentence's categories. More specifically, the "*claim_lexicon"* was created including the words ['reveal', 'provide', 'confirm', 'suggest','Reveal', 'Provide', 'Confirm', 'Suggest', 'in conclusion', 'In conclusion','conclusion','Conclusion','conclusions','Conclusions'] and the "*evidence_lexicon"* including the words ['result','results','Result','Results','findings','Findings'].

By doing so, when we spotted a sentence that contained a word that belonged to either of the lexicons, then it was labeled accordingly. Secondly, we decided to label the last sentence of every abstract as a claim sentence. This decision was made due to the fact that the majority of the abstracts during our annotation task followed this pattern.

After all these rules where applied, an additional label column was created to our dataframe in order to identify how well our rules behave with regards to the correct labeling. In order to achieve that we used the following metrics – statistics:

- $\text{Recall} = \frac{True\ positive}{True\ positive + False\ Negative}$
- $\text{Precision} = \frac{True\ positive}{True\ positive + False\ Positive}$
- $\text{F} - \text{score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

For classification tasks, the terms true positives, true negatives, false positives, and false negatives compare the results of the classifier under test with trusted external judgments. The terms positive and negative refer to the classifier's prediction, and the terms true and false refer to whether that prediction corresponds to the external judgment (sometimes known as the observation).

The values of the aforementioned metrics are presented in Table 1.

	Recall	Precision	F- Score
Claim	0.594	0.470	0.525
Evidence	0.084	0.333	0.135

Table 1: Recall, Precision and F-Score of the baseline model's labelling

6 Data pre-processing steps

6.1 Data cleaning

Data cleaning and preparation of the raw text is an essential step before fitting any machine learning or deep learning model. For the data pre-processing, we used the Natural Language Toolkit (NLTK), a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

As a first step, we defined a function called "clean_sentence" that applied the following processes to each sentence. First, it removed any punctuation marks from the sentence. This was achieved with the use of RegEx (Regular Expressions) from the 're' Python module. A RegEx is a special sequence of characters that uses a search pattern to find a string or set of strings. It can detect the presence or absence of a text by matching with a particular pattern, and also can split a pattern into one or more sub-patterns. Then, the function lower cased the words in the sentence. In the next step, it removed any English Stopwords, commonly used words (such as "the", "a", "an", "in") that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. Then, it lemmatized the tokens and it removed from the sentences characters which are not digits.

Since our problem is a multi-class classification problem, we needed to pass our model a three-dimensional output vector.

6.2 Splitting data into train and test sets

Having applied any data cleaning procedures, the dataset was splitted into train and test sets. We used 70 % of the data for training and 30 % for testing. Random state so that every time we get the same training and testing data. Also, stratification was applied to the label in order to get an even percentage of the labels, which is the target class, as in the train dataset.

In order to apply the text classification, we tokenized the data into a format that can be used by the word embeddings. Keras offers a couple of convenience methods for text preprocessing and sequence preprocessing which we can employ to prepare your text. We used the Tokenizer utility class which can vectorize a text corpus into a list of integers. Each integer maps to a value in a dictionary that encodes the entire corpus, with the keys in the dictionary being the vocabulary terms themselves. We added the parameter num_words, which is responsible for setting the size of the vocabulary. The indexing is ordered after the most common words in the text. It is important to note that the index 0 is reserved and is not assigned to any word. This zero index is used for padding, which we will implement afterwards.

A common faced problem is that each text sequence has in most cases different length of words. In order to encounter this, we used the method pad_sequence() which simply pads the sequence of words with zeros. By default, it prepends zeros but we want to append them. Furthermore, in order to specify how long the sequences should be, we added a maxlen parameter, that cuts sequences that exceed the defined number.

7 CNN Models

7.1 First CNN Model

7.1.1 Model Description

By using the Keras Python library for developing and evaluating deep learning models, we introduced a sequential model, which is a linear stack of layers, and we added as an input layer the Embedding Layer of Keras, which takes the previously calculated integers and maps them to a dense vector of the embedding. The embedding layer used as input the size of the vocabulary, multiplied with the length of the sequence and gave as an output the word embeddings of size of the dense vector. Then we normalized the output of the embedding by using a dropout layer. Then we added a convolutional layer using a number of filters equal to 32, kernel size equal to 3, by using as activation function the ReLU function.

Afterwards, we added a pooling layer as a way to reduce the size of the incoming feature vector. In our case we used a GlobalMaxPooling1D, which takes the max of the maximum value of all features in the pool for each feature dimension. Then we added a dense layer with the ReLU activation function and we normalized the output by using again a dropout layer. Finally, given that our problem is a multi-class problem we added a dense layer with the size of the number of our classes with activation function = softmax .

Before starting with the training of our model, we had to configure the learning process. This was achieved with the *compile()* method which specifies the optimizer, the metric function and

the loss function. Metric functions are similar to loss functions, except that the results from evaluating a metric are not used when training the model. In our case, we used the 'adam' optimizer, the categorical crossentropy as a loss function and the categorical accuracy combined with the mean absolute error as a metric.

We started training our model by utilizing the *fit()* function. Given that in neural networks the training constitutes an iterative process, we had to specify the number of completed iterations, called epochs, we wanted the model to be trained. We run it for 100 epochs. We also specified the batch size, which is responsible for the number of samples that we want to use in one epoch, to be equal to 64. Furthermore, we added an early stopping callback using the validation loss as a criterion to stop training the model.

7.1.2 Results and Quantitative Analysis

After completing the model's training, we evaluated the performance of our model by using the *evaluate()* method and the accuracy metrics of the mean absolute error and the categorical crossentropy.

The loss, the mean square error and the accuracy of the 1st CNN model, in both the training and the test datasets, are depicted in the following Figures, (Figure 3 to Figure 5).

The history callback was used in order to visualize the following graphs. As it observed, the 1st CNN model does not present the same performance in the training and the test datasets.

It seems as if we have trained our model for too long since the training set reached 100% accuracy. A good way to see when the model starts overfitting is when the loss of the validation data starts rising again. This tends to be a good point to stop the model. We can see this around 3-4 epochs in this training.

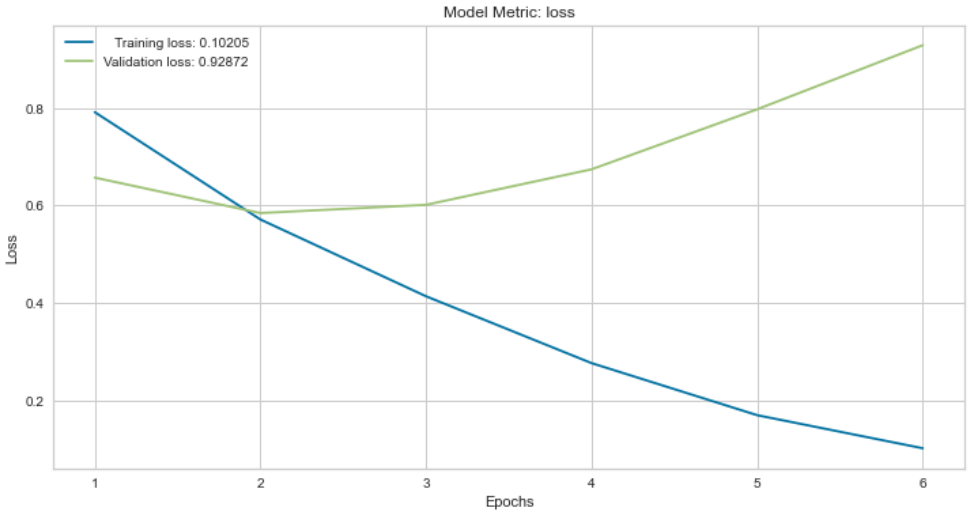

Figure 3: CNN model's loss per epoch

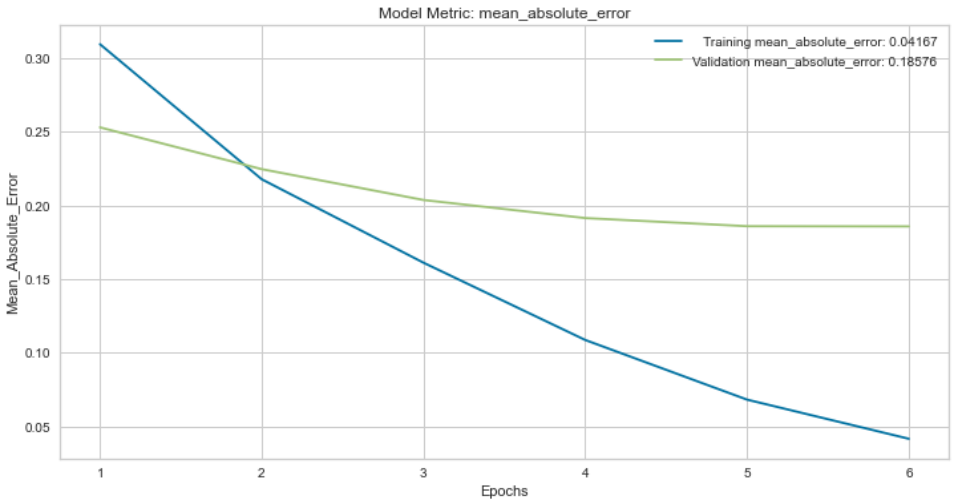

Figure 4: CNN model's mean absolute error per epoch

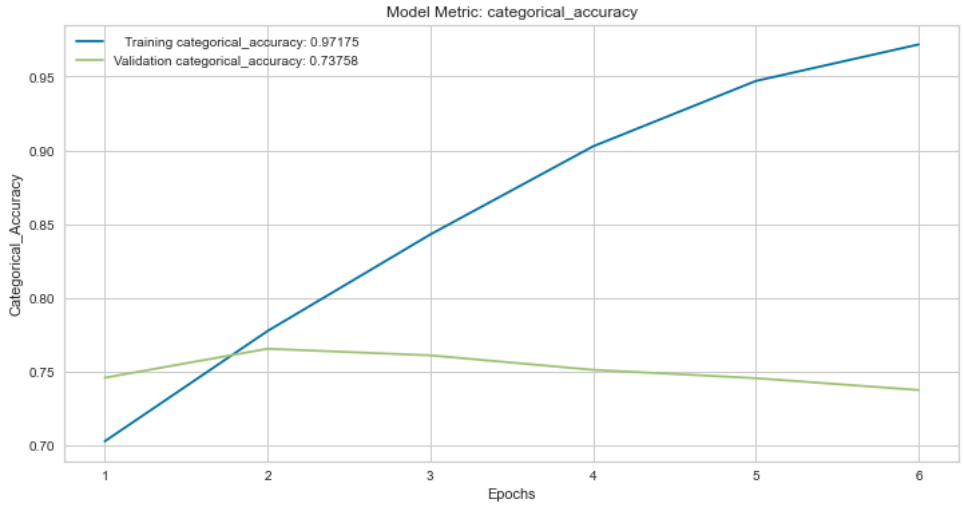

Figure 5: CNN model's accuracy per epoch

Furthermore, the Confusion Matrix of the first CNN Model, indicating the actual and the predicted classified sentences, is presented in Figure 6. The rows represent the actual labels, whereas the columns the predicted labels. As it is discerned, 1,018 sentences were predicted correctly as evidence, 250 sentences were actual claims that the model has successfully predicted and 6094 sentences were correctly predicted as no-label. These numbers are considered as satisfactory. However, the fact that the majority of the wrongly predicted claims and evidence sentences, were classified as no-label is not satisfactory.

	0	1	2
0	6094	491	128
1	797	1018	48
2	594	182	250

Figure 6:Confusion Matrix of 1st CNN Model

The precision, the recall and the f-score which were described in Section 5 of the first CNN model are presented in Figure 7. The recall of the claims and evidences is equal to 24% and 55% respectively. The accuracy of the model, equal to 76.67%, indicates that the performance could be improved.

	precision	recall	f1-score	support
0	0.8142	0.9078	0.8584	6713
1	0.6020	0.5464	0.5729	1863
2	0.5869	0.2437	0.3444	1026
accuracy			0.7667	9602
macro avg	0.6677	0.5660	0.5919	9602
weighted avg	0.7487	0.7667	0.7481	9602

Figure 7:Clasification Report of the 1st CNN Model

Also, we plotted the ROC Curve, a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: the True Positive Rate and the False Positive Rate. The plot in Figure 8, includes the curves representing the general predictions of the 1st CNN model, but also the curves that represent each class's predictions.

The macro-average ROC Curve computes the metric independently for each class and afterwards takes the average, whereas the micro-average ROC Curve aggregates the contributions of all classes to calculate the average metric. The aforementioned metrics were equal to 84% and 91% respectively. Given that a relatively small difference is presented in the metrics of each class, it can be assumed that this model classifies in a satisfactory level individual class.

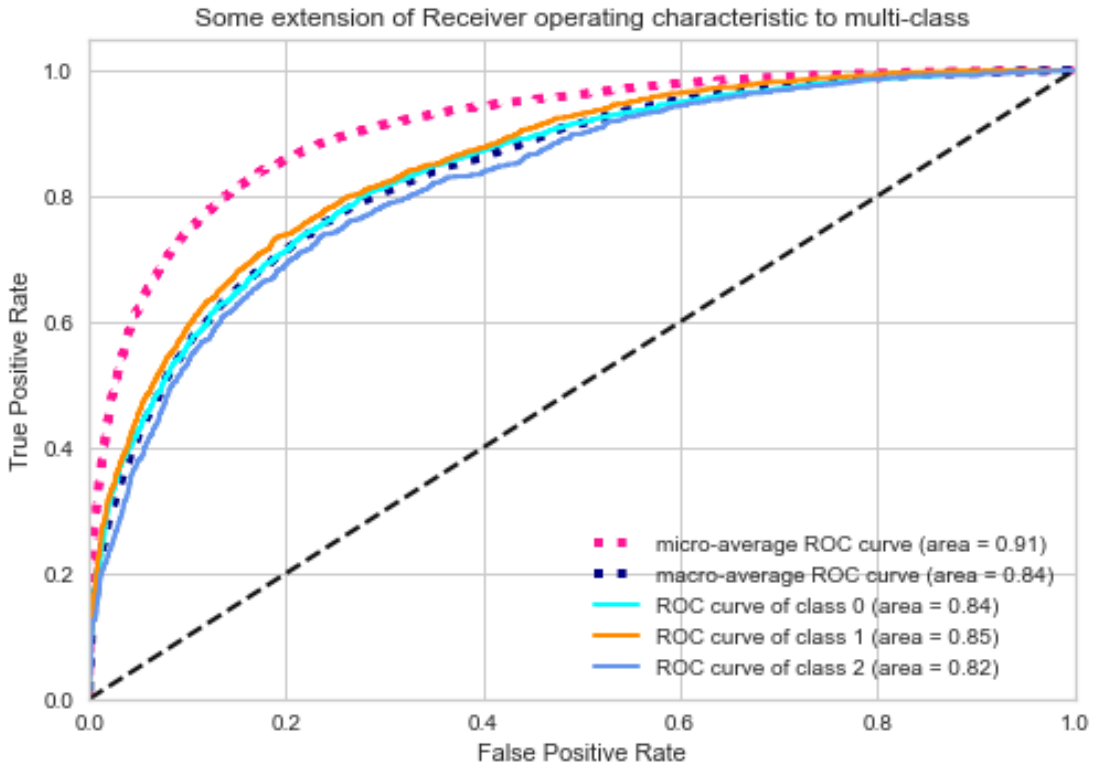

Figure 8: 1st CNN model's ROC Curve

7.2 Second CNN Model

7.2.1 Model Description

Apart from the model that we constructed, we also run a cnn model based on the word embeddings method, which represents words as dense word vectors that are trained. The aim of this method is to map semantic meaning into a geometric space, the embedding space, implying that word embeddings do not conceive the text as humans would do, but they map the statistical structure of the language used in the corpus. We used a precomputed embedding space. It is possible to precalculate word embeddings by training them on a large corpus of text.

Word2Vec, developed by Google, and GloVe (Global Vectors for word representation), developed by Stanford NLP group are the most popular methods. Although the Google model performs better, it demanded more memory. Due to this restriction, we decided to use the GloVe word embedding which includes 400,000 lines. Each line represents a word along with vectors as a stream of floats. GloVe precomputes word embeddings with a co-occurrence matrix and by using matrix factorization.

We checked how many of the embedding vectors are non-zero. 56.8% of the vocabulary is covered by the pre-trained model, which is a satisfactory coverage of our vocabulary. We also allowed the embedding to be trained by using trainable = True.

7.2.2 Results and Quantitative Analysis

The loss, the mean square error and the accuracy of the 2nd CNN model, in both the training and the test datasets, are depicted in the following Figures, (Figure 9 to Figure 11).

As it observed, the 2nd CNN model does not present the same performance in the training and the test datasets.

It seems as if we have trained our model for too long since the training set reached 100% accuracy. We can see this around 3-4 epochs in this training.

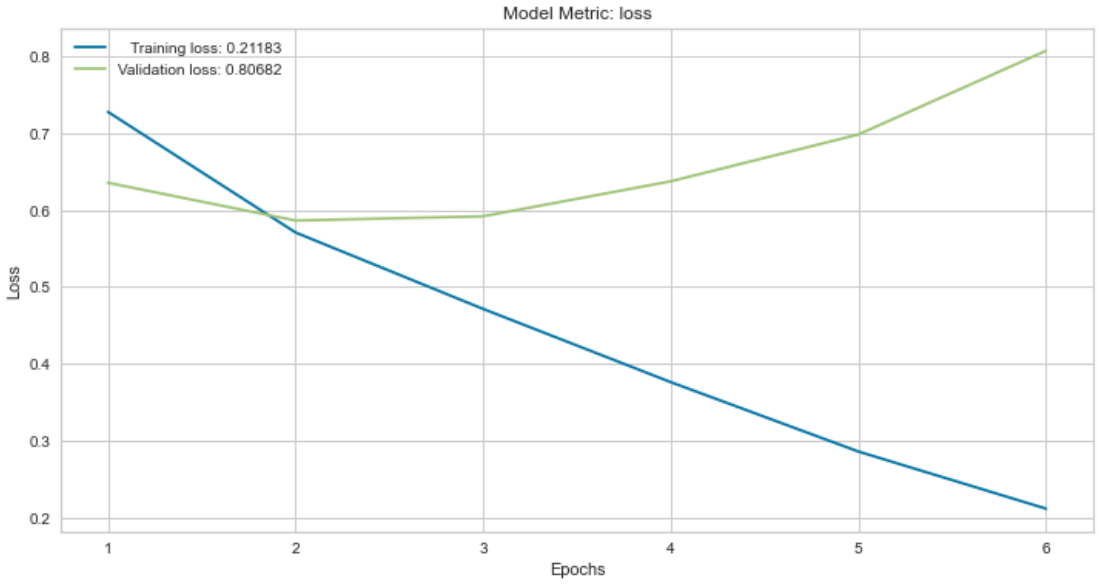

Figure 9: CNN model's loss per epoch

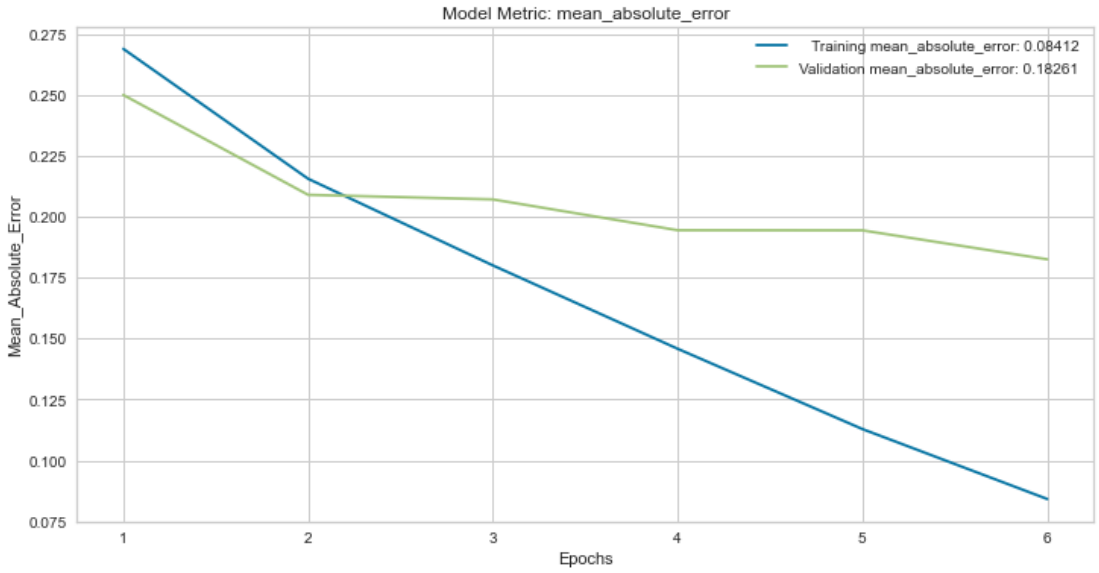

Figure 10: *CNN model's mean absolute error per epoch*

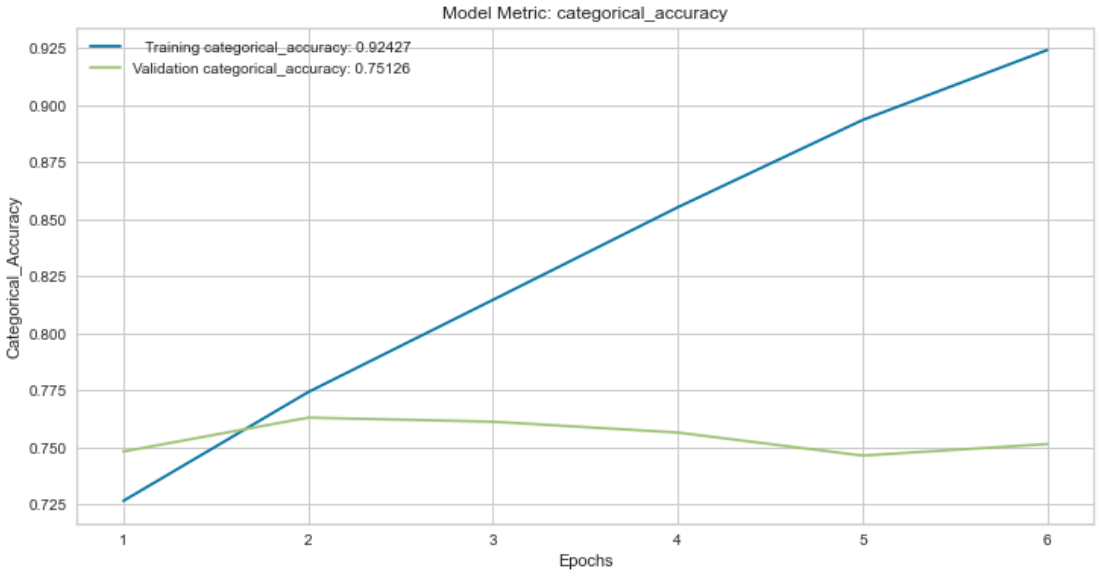

Figure 11: CNN model's accuracy per epoch

Also, the Confusion Matrix of the 2nd CNN Model, indicating the actual and the predicted classified sentences, is presented in Figure 12*Figure 6*. As it is discerned, 820 sentences were predicted correctly as evidence, 262 sentences were actual claims that the model has successfully predicted and 6304 sentences were correctly predicted as no-label. These numbers are considered as satisfactory. However, the fact that the majority of the wrongly predicted claims and evidence sentences, were classified as no-label is not satisfactory.

	0	1	2
0	6304	284	125
1	994	820	49
2	666	98	262

Figure 12:Confusion Matrix of the 2nd CNN Model

The precision, the recall and the f-score of the second CNN model are presented in Figure 13*Figure 7*. The recall of the claims and evidences is equal to 26% and 44% respectively. The accuracy of the model, equal to 76.92%, indicates that the performance could be improved.

	precision	recall	f1-score	support
0	0.7916	0.9391	0.8590	6713
1	0.6822	0.4402	0.5351	1863
2	0.6009	0.2554	0.3584	1026
accuracy			0.7692	9602
macro avg	0.6916	0.5449	0.5842	9602
weighted avg	0.7500	0.7692	0.7427	9602

Figure 13: Clasification Report of the 2nd CNN Model

Also, we plotted the ROC Curve, which is presented in Figure 14, includes the curves representing the general predictions of the 2nd CNN model, but also the curves that represent each class's predictions.

The macro-average ROC Curve and the micro-average ROC Curve were equal to 84% and 91% respectively, the same as in the previous model. Given that a relatively small difference is presented in the metrics of each class, it can be assumed that this model classifies in a satisfactory level individual class.

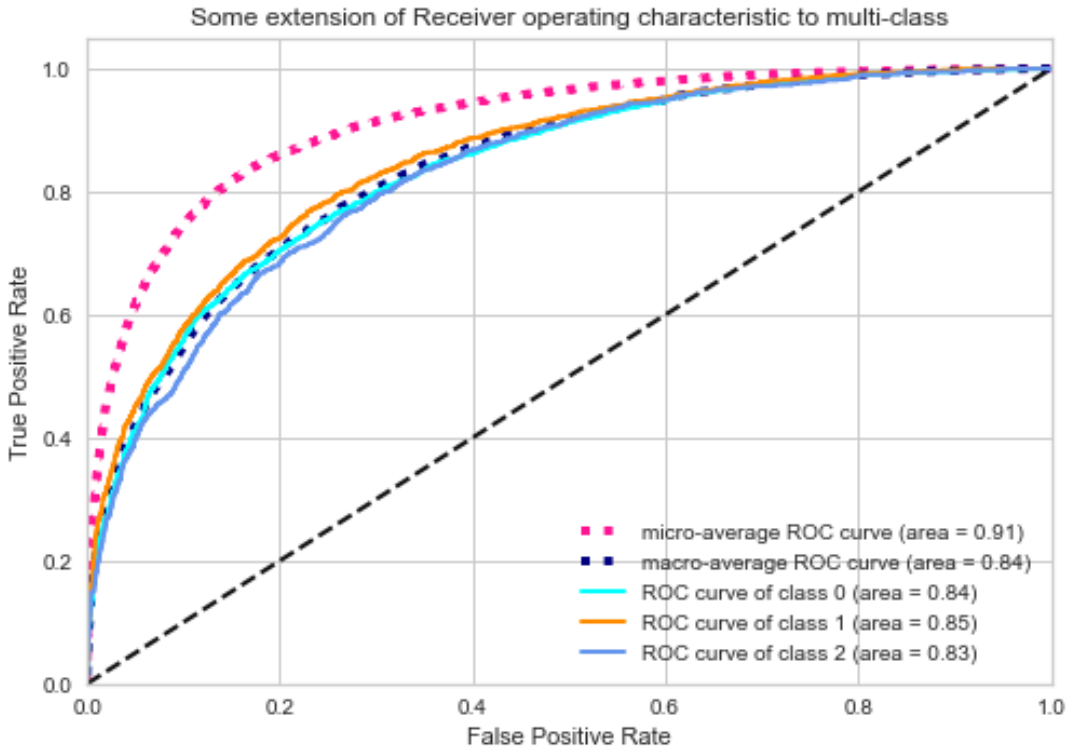

Figure 14: 2nd CNN model's ROC Curve

8 Clustering

As far the clustering process that was followed is concerned, we approached this issue using K-Means algorithm. We conducted two types of clustering, the first one was based on document embeddings about the whole abstract and the second one was based on sentence embeddings for the Claim labels only.

8.1 Abstract-Custom Model

With regards to our first type of clustering (the whole abstract) we firstly generated document vectors by using the custom model (based on Word2Vec) that was created-trained earlier.

We get all the word vectors of each abstract and average them to generate a vector per each abstract. Those document vectors were used as input features during our clustering. In order to find the appropriate number of clusters (K), we used the Silhouette coefficient as an indicator and thus resulted to K=2 number of clusters.

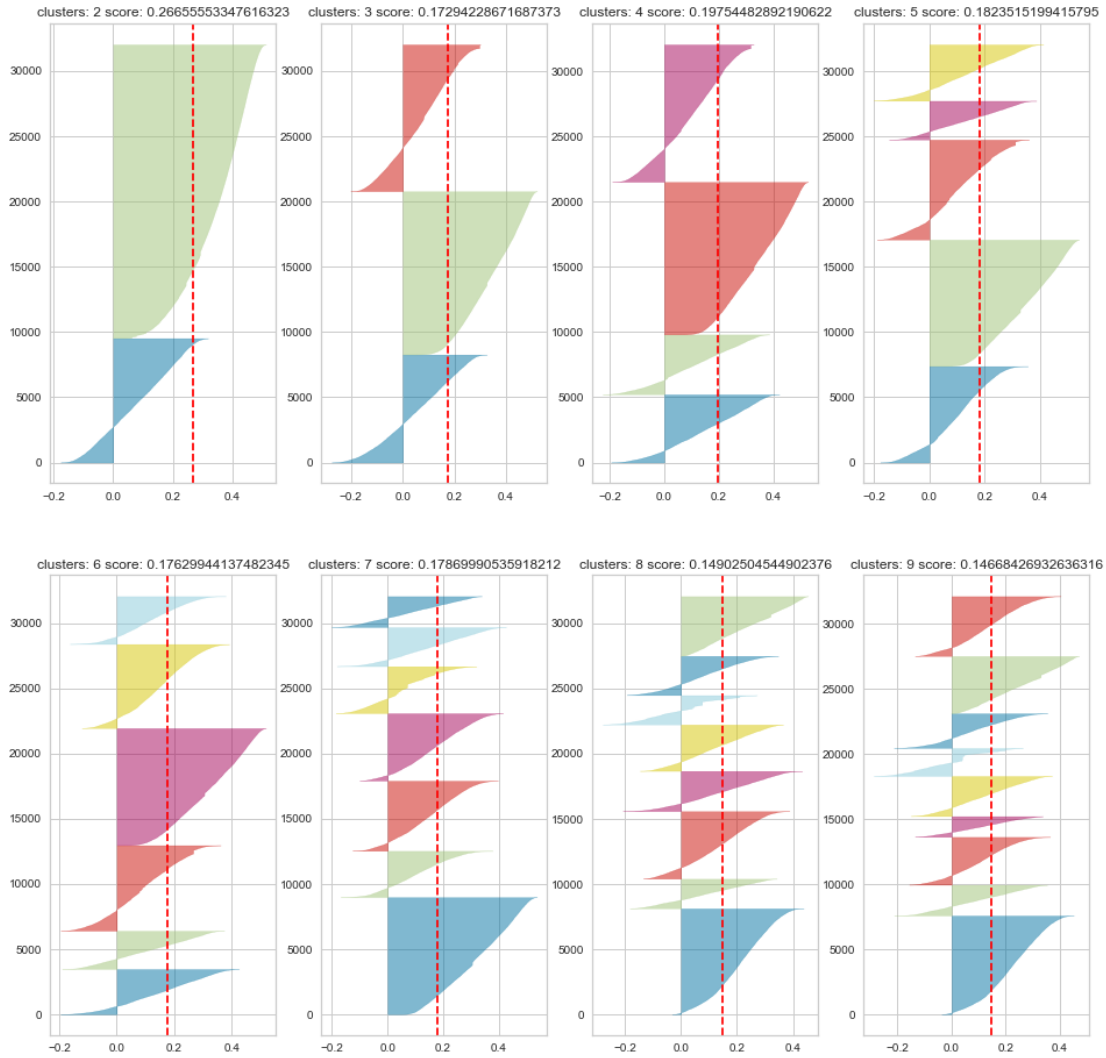

Figure 15:

8.2 Abstract-Pretrained Embeddings

Similarly, we followed the same approach as mentioned before, but this time we also used trained pre-calculated word embeddings, more specifically we used the GloVe method. Through this method we get 400000 word vectors with 300 elements size per vector and they were used as features during our clustering. Again, the instantiation of the number of clusters was based on the Silhouette coefficients and this time the ideal number of clusters was equal to 3 (K=3)

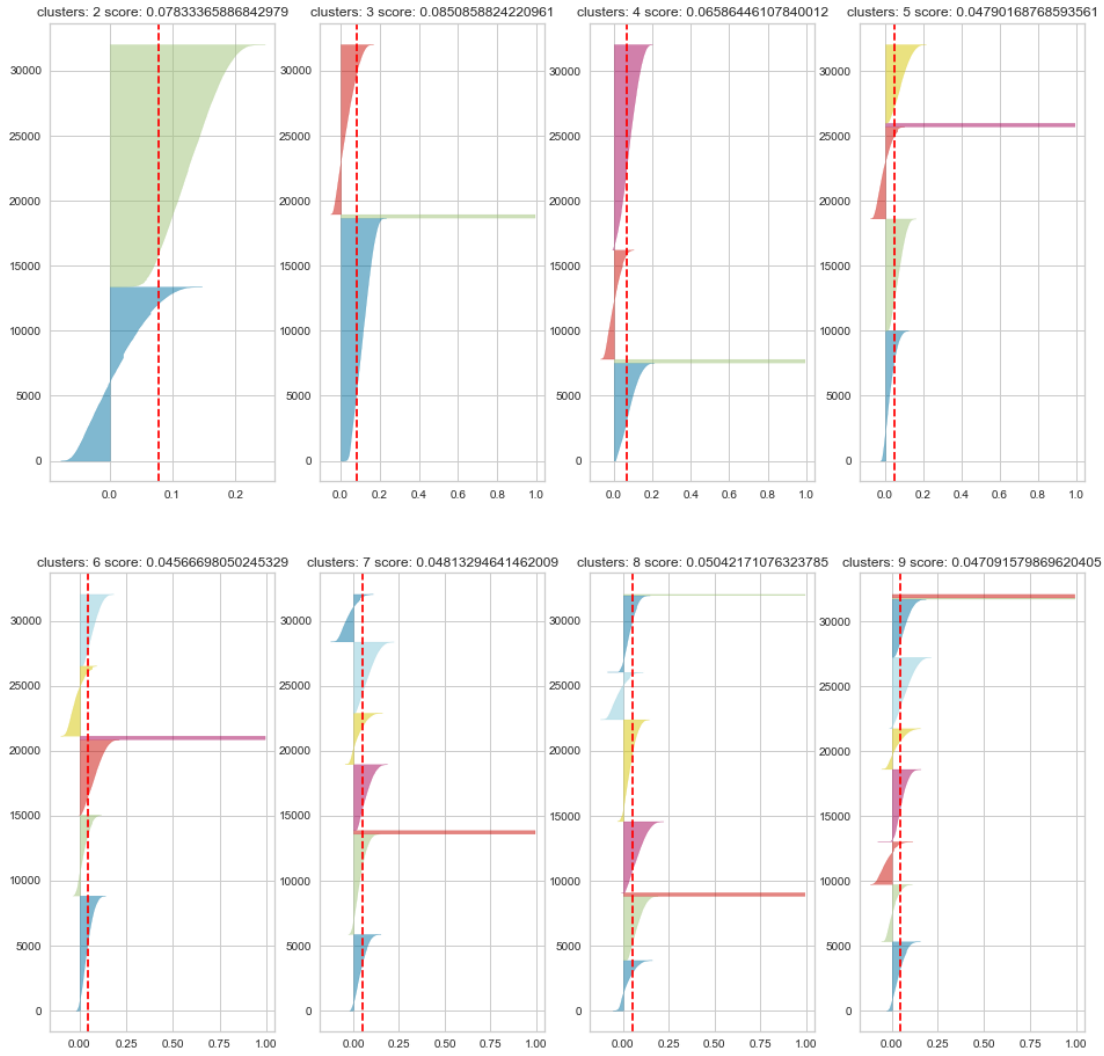

Figure 16:

8.3 Claim-Custom Model

Regarding our second type of clustering (claim only), we isolated only the claim sentences by filtering our dataframe. Again, we firstly created our sentence embeddings by using our already created custom model (model_t). After that, they were feed to our clustering as features and we again used Silhouette coefficient to find our number of clusters K=2

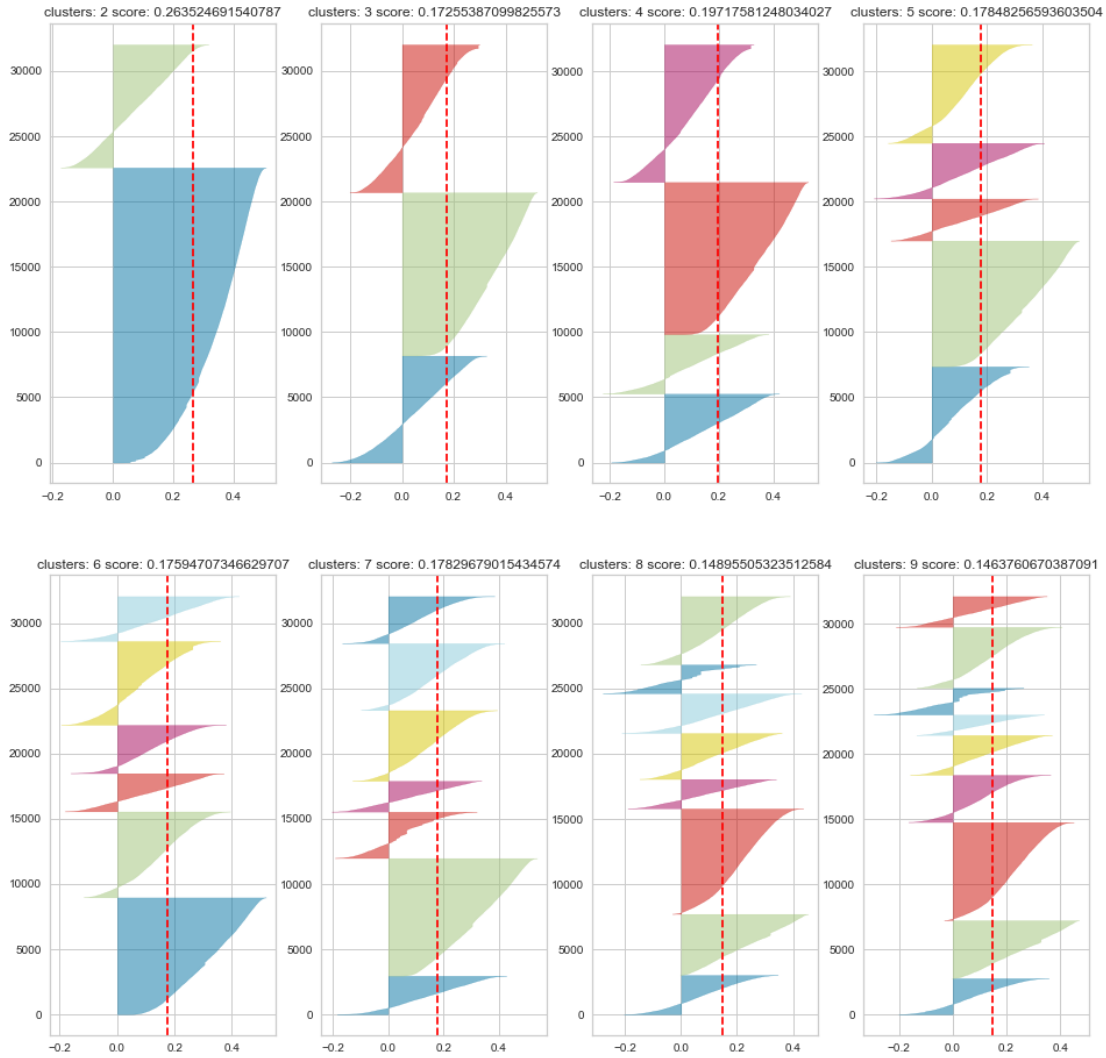

Figure 17:

8.4 Claim- Pretrained Embeddings

The same procedure as described above, but the GloVe pre-trained word embeddings were used instead. Based on the Silhouette coefficient as an indicator in order to find the appropriate number of clusters (K), we resulted to K=3 number of clusters.

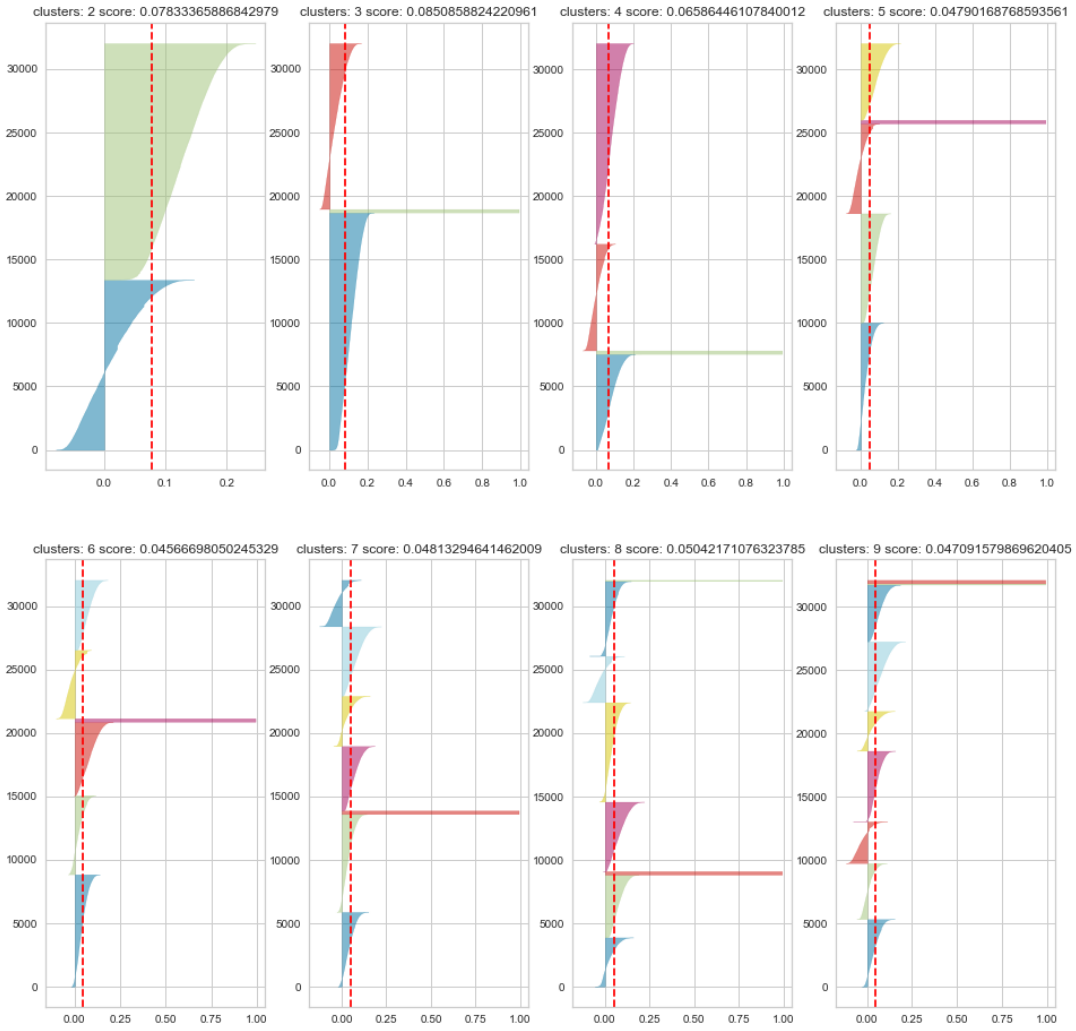

Figure 18:

9 Discussion, Comments/Notes

In order to check how good our CNN models we reproduced a dummy classifier based on the frequency of our labels. The accuracy of the dummy classifier was 69% while the accuracy of both of our CNN models was 77%.

With regards to our clustering results, we can observe that our silhouette values are close to 0 and this means that the centroids of the clusters are very close to one another. As a result we can infer that our clustering is not so good, as the clusters are not well defined.

10 Members/Roles

The team consisted of the following three members, all attending the MSc in Business Analytics of Athens University of Economics and Business:

- Maria Anna Kasma, graduate of the Department of Applied Mathematics and Physical Sciences of the National Technical University of Athens
- Ilias Tsekouras, graduate of Department of Economics of the Athens University of Economics and Business,
- Spiros Tsiolis, graduate of Department of Economics of the Athens University of Economics and Business

Regarding the roles of each team member, no specific role was assigned to them. All of them were involved in the project's annotation phase, contributing to the formation of the final dataset, which was provided to all teams that chose to pursue with the pilot project. Having acquired the final dataset, again all three members discussed about the data pre-processing, the general conceptual description of the applied methodology. Maria Anna Kasma and Spiros Tsiolis dealt with the baseline model. The construction of the two CNN models was processed by Ilias Tsekouras and Spiros Tsiolis. Furthermore, all members were involved in the abstracts' clustering as well as in the discussion of the models' performance. Finally, the synthesis of the report and the preparation of the project's presentation was the result of teamwork as well.

11 Time Plan

The Time Plan of the Argument Mining is designated in *Figure 19:*Figure 19. In general, we can assume that it was splitted in the following Phases:

- **1st Phase:** Decision of project
- **2nd Phase:** Annotation Phase
- **3rd Phase:** Model Implementation
- **4th Phase:** Delivery of report and Presentation

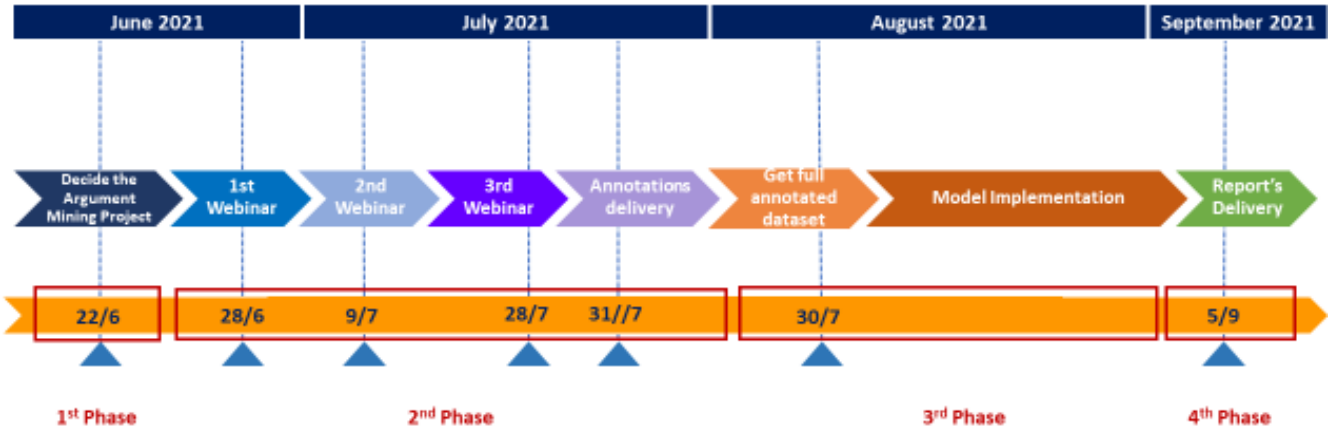

Figure 19: Milestones of Argument Mining Project

After having decided that we as a team we would follow the pilot project instead of a project of our choice, we attended the first webinar in which we discussed with Mr. Fergadis the annotation guide as well as the annotator agreement. At the end of the webinar, we were given a small dataset of abstracts (Set A) that we had to annotate until the 2nd webinar, in order to apply what we were taught in that 1st webinar. The aim of the 2nd webinar, was to resolve any problems that arised during the annotation of Set A and to gain feedback about the annotation agreement. Until the next webinar, we checked as a team each abstract that we had annotated independently as team members, in order to discuss any deviations and possible errors occurred during the annotation phase. This was an essential to be discussed as a paper could be considered as valid only if two out of three team members' annotation labelling matched. During the 3rd Webinar, we discussed about the different methods we could apply in order to construct our deep learning models. On 23 of July, we had to deliver our annotations. This was extended to one week later. In the beginning of August we got the full dataset and we were ready to start with the model implementation phase which included the programming part of the project, concerning the data pre-processing and the training of a few models. After the creation of the first models, we discussed the result and any possible solution that would improve their performance. Hence, a fine tuning of the models' parameters was applied. Then,

the preparation of the report as well as the presentation required around a week to be completed and submitted within the project's extended deadline on the 5th of September.

12 Bibliography

http://ceur-ws.org/Vol-2847/paper-03.pdf

https://www.ijcai.org/Proceedings/15/Papers/033.pdf

Annotation Guide, Aris Fergadis, June 2021