Abstract Mining Assignment

Machine Learning and Content Engineering

MSc in Business Analytics

Kanellou Ioanna F2822005

Mavridis Georgios F2822007

Papantoni Evangelia F2822013

September 2021

Table of Contents

Scope

The scope of this assignment was initially to manually assess and mark the structure and argument components of multiple abstracts and upon consolidation of the results, to create a machine learning model in the context of argumentation mining from multiple abstracts provided. Also, we created clusters from the documents based on their content (document, claim & evidence embeddings) combined with other features such as project objective and EU call.

1. Introduction

Over the years, the research area has evolved due to technological innovations, so researchers had to expand the way that they see the science. This was difficult because all the documentation about a lot of sciences had to be changed on a theoretical as well as practical level. This change not only influences the researchers, but also the data analysts and data engineers that want to find a pattern between the documentation. ML engineers and data analysts always try to find an argumentation between papers from a lot of research areas. This analysis is called argumentation. Argumentation is the act or process of forming reasons and of drawing conclusions and applying them to a case in a discussion or research.

More specifically, argumentation is the interdisciplinary study of how conclusions can be reached through logical reasoning.

Moreover, Argumentation is a very complex phenomenon, and argument mining is just an approximation / simplification of some aspects of argumentation analysis in order to extract prototypical arguments or parts of them with a reasonably good accuracy. Some key components of the argumentation are given below:

- Understanding and identifying arguments,
- Identifying the points that the conclusion is drown, and
- Establishing the proof in the sentence.

The key point in the argumentation is the argument mining. Argument mining is the automatic identification of the structure of the sentence and reasoning expressed as arguments in the natural language.

Argument mining also depends on the goal that is pursued, for example:

- Finding argumentative statements;
- Finding supporting and objecting statements;
- Derive the structure of arguments;
- Assessing argumentation.

In this task we focus on finding argumentative statements. More specifically, an argumentative statement takes an opinion, asserting the writer's stance. Questions, or vague statements or quotations are not argumentative statements because they do not explain the writer's opinion.

2. Our project

In this project, the first task was to manually do the argumentative analysis, considering the steps that have to be done, the available tools that we had and the limitations of time and papers. That is because, understanding the argument analysis manually, gave us an insight about the papers and the analysis. More specifically, manually analysis offers a unique insight into the task and provides the analysts valuable insights, in order to find a useful and best fit way to automate the task and to unpick the complex argumentative relationships represented in the natural language. Moreover, the argumentative structure is in the text, but it can be found not only manually with a pen and a paper but also with the use of simple graphics software with diagramming tools as we use in the analysis of this project. The advantages of this approach are multiple, first of all gives an insight about the data and second the data analysts use it in order to see the insight of the argumentation and then find an automate model to do the task.
Moreover, this manually argumentative analysis was used to create the dataset that all the teams used in order to automate the task. The dataset was already ready and the teams didn't need to manipulate it. The second task was to create a simple baseline based on the exploratory data analysis. This analysis is used in order to examine the structure of the data such as the labels distributions in the abstracts and the arguments. Furthermore, after the exploratory data analysis, the task was to use one of the three main approaches that the project gives in order to automate the argumentative analysis. The three approaches were the FastText, the custom and the transformer network approach. More specifically, the Custom approach gets as input the abstract and the sentences but with the preprocessing that the data analyst gives to the model such as the tokenization, lemmatization, bag of words etc. The custom approach can be run by a CNN or RNN model which are unimodal or multimodal models with shared embedding layer. The FastText approach will be analysed below, because in this project this approach is being used. The transformer network approach is used for the arguments and for the structure labels. The last task in the project was to do an abstract clustering either by using the clustering models that are in the bibliography such as k-means, hierarchical clustering, or by using graphs.

3. Our Vision/Goals

Our goal is to present a case study for automatic argumentative text classification. The texts are taken from papers that are available in the bibliography and the internet. The presented classification scheme in this study is based on Natural Language Processing that do not require the calculation of any hand-engineered text features.

On this report, we will try to cover the annotations monitoring through text classification processes and models. Although the presented approach is based on papers and texts that are already be prepared, the model that is represented it can be used in several other cases of text classification.

4. Methodology

The goal of the text classification is to assign documents to one or multiple categories such as claim or evidence. Nowadays, the most popular approach to build such classifiers is machine learning. The machine learning approaches need labeled data, which consists of documents and their categories such as labels or tags. The architecture of the model build for text classification was a FastText approach. FastText is an open-source, free library that allows users to learn text representations and text classifiers. It works on standard general hardware such as python that was used in this project.

In order to achieve the goal of building an efficient machine learning model that would effectively perform argumentation mining with increased prediction results the following approach was followed:

A simple baseline model based on exploratory data analysis was created to compare it with the FastText model. Through the utilization of FastText, we trained a supervised FastText model with various input parameters of the model based on the argument characteristics of the abstracts. Therefore, through the autotune approach, we determined the hyperparameters that provided the best prediction results. Upon the creation of the necessary document embeddings and tokenization for the abstracts and the project objectives, the creation of the sentence embeddings for the claim arguments were created. This allowed us to perform an abstract clustering, through the application of a k-means clustering algorithm with the use of only the abstracts' document embeddings, the use of abstracts' document and project objective embeddings and lastly with the use of abstracts' document and claim embeddings. More specifically, K-means clustering is an unsupervised machine learning algorithm which is the most simple and popular. The unsupervised algorithms make inferences from datasets using only input vectors without referring to known or labelled outcomes. The k-means algorithm in order to run has to be defined a k number of clusters through reducing the in-cluster sum of squares. The items in the same cluster have to be similar and items from different clusters need to be different in order to have the best fit cluster. The k-means works as follows: the mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and through iterative calculations the centroids are optimized. Finally, when the centroids have stabilized and there is no change in their values, the defined values of the clustered are represented.

5. Data Collection

The research and the collection of the data is considered the most important part of the implementation of a Deep Learning model.

In this project, the dataset was created by all the teams. As it is described above, the first task was to manually do the argumentative analysis in a software that the professor gave to us. All sub-teams had a specific number of annotations and time in order to fulfill manually the argumentative analysis. Moreover, argumentation guidelines were provided in order for the argumentative analysis to be performed and for the sentences of the annotations to be categorized into claim or evidence. Upon finalization of the manually argumentative analysis which contained the abstracts' labeling, the data were accumulated in a single dataset for further processing and analysis with the use of deep learning models. The final dataset was suitable to text classification.

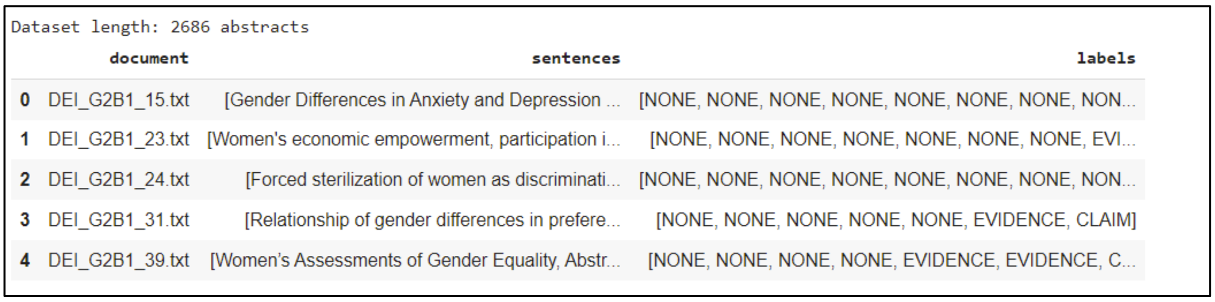
6. Dataset Overview

Prior proceeding with the approach that we would follow for the creation of our argumentation mining model, we were aware that our dataset would be comprised from the following components:

- Structure:
    - Background;
    - Objective / Aim;
    - Method;
    - Result;
    - Conclusion;
- Argument:

- Evidence;
- Claim;
- Neither.

The dataset that was imported in the Jupiter notebook to be manipulated and explored was comprised by a dataset provided by the professor and a dataset that we created by annotating the provided abstracts. The accumulated dataset contained 2686 abstracts.



```
Dataset length: 2686 abstracts
```

	document	sentences	labels
0	DEI_G2B1_15.txt	[Gender Differences in Anxiety and Depression ...	[NONE, NONE, NONE, NONE, NONE, NONE, NONE, NON...
1	DEI_G2B1_23.txt	[Women's economic empowerment, participation i...	[NONE, NONE, NONE, NONE, NONE, NONE, NONE, EVI...
2	DEI_G2B1_24.txt	[Forced sterilization of women as discriminati...	[NONE, NONE, NONE, NONE, NONE, NONE, NONE, NON...
3	DEI_G2B1_31.txt	[Relationship of gender differences in prefere...	[NONE, NONE, NONE, NONE, NONE, EVIDENCE, CLAIM]
4	DEI_G2B1_39.txt	[Women's Assessments of Gender Equality, Abstr...	[NONE, NONE, NONE, NONE, EVIDENCE, EVIDENCE, C...

Figure 1:Argument dataset overview

7. Data Processing/Annotation/Normalization

Baseline model

We separated the dataset in training and testing and we used the training set to a dataframe only with claims and another with evidences. Then we found the 50 most common words for each dataframe.

FastText model

The first step in the manipulation of the dataset was to split the documents, for the model the sentences had to be separated. The manipulation was to keep the same document index in a new column in order to re-group the sentences to a document. Another step was to convert the column sentence from object to string and to remove any blanks from the start and the end of the new string. The next steps of the manipulation were to split the dataset into training, validation and testing datasets in order to test if the model is accurate. Looking at the data, some words contained uppercase letter or punctuation. One step to improve the structure of the dataset in order to help the training of the model was to apply some simple pre-processing using genism library which converts a document into a list of tokens, lowercases and tokenizes the sentences. Also, we add the _label_ prefix to every label, which is how FastText recognize the labels. Finally, we converted the training, validation and test sets to txt format and we were ready to start the text classification algorithm which will give us the best fit model that is represented below. It should be noted that we followed the same approach for the dataset with the structure labels.

Clustering

Firstly, we used Keras tokenizer to create document embeddings from the abstracts. We only considered the 15K most used words in this dataset and we used Keras pad sequences tool to adds zeros to sequences that are too short. The embeddings will consist for 500 words. Also, we extracted project objective embeddings of size 100 using the Keras tokenizer and pad sequences. Then we combine the document embeddings and project embeddings to a dataframe. Finally, we followed the same approach from above for the creation of claim embeddings and we create a dataframe only with claim embeddings and we merged it with the document embeddings.

8. Algorithms, NLP architectures/systems

Baseline model

Based on our exploratory analysis, we created a list with words that was used in order to classify a sentence as claim. We also create an additional list with words from the evidences. The next step was to create an algorithm which will take every sentence from the test dataframe and will check if it contains a word that belongs in one of the two lists. If there is a word in the claim list, we will classify the sentence as claim. If the word belongs in the evidence list, we will classify the sentence as evidence. Otherwise, we assign to the sentence the label Neither.

FastText model

FastText is an open-source library developed by the Facebook AI Research lab that provides pre-built models for text classification both supervised and unsupervised. Its main focus is on achieving scalable solution for tasks of text classification and representation while processing large datasets quickly and accurately. The core of FastText library relies on the Continuous Bag of Words (CBOW) model for word representation and a hierarchical classifier to speed up training. Continuous Bag of Words (CBOW) is a shallow neural network that is trained to predict a word from its neighbors. FastText replaces the objective of predicting a word with predicting a category. These single-layer models train incredibly fast and can scale very well. Also, FastText replaces the softmax over labels with a hierarchical softmax. It should be noted that each node represents a label. This reduces computation as we don’t need to compute all labels probabilities, and the limited number of parameters consequently reduces training time. Therefore, the FastText classifier was used to classify the quality of the annotations performed to the papers of our dataset.

9. Experiments – Setup, Configuration

Baseline model

The baseline model is the basic model of our assignment that was created for comparison with the FastText model. Therefore, no particular configurations were performed at this stage.

FastText model

The FastText model needs labeled data to train a supervised classifier. So, each line of the text file contains a list of labels, followed by the corresponding document. All the labels start by the _label_ prefix, which is how FastText recognize what is a label or what is a word. Before training the first classifier, the data had to be splitted into train and validation. The validation dataset is used to evaluate how good the classifier learned and to optimize the hyperparameters of the model.

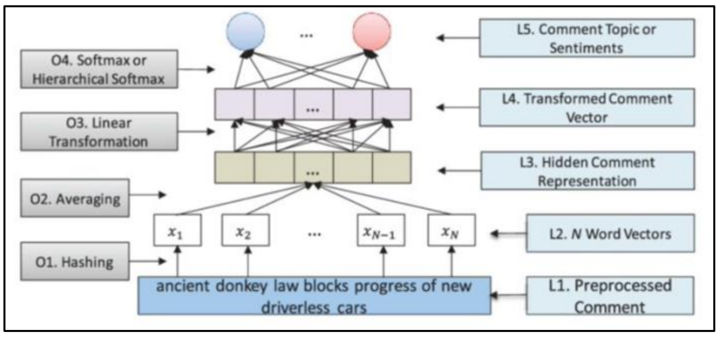

O4. Softmax or Hierarchical Softmax
O3. Linear Transformation
O2. Averaging
O1. Hashing
L5. Comment Topic or Sentiments
L4. Transformed Comment Vector
L3. Hidden Comment Representation
L2. N Word Vectors
L1. Preprocessed Comment
x_1
x_2
x_{N-1}
x_N
ancient donkey law blocks progress of new driverless cars

Figure 2: Model Architecture of FastText where L represents Layer and O operation

By default, FastText sees each training example only five times during training. The number of times each example is seen is known as the number of epochs and can be increased or decreased by using the -epoch option. Another way to change the learning speed of our model is to increase or decrease the learning rate of the algorithm. This means how much the model changes after processing each example. More specifically, a learning rate of 0 would mean that the model does not change at all so does not learn anything. In theory, good values of the learning rate are in the range 0,1-1. Finally, by using the word bigrams, instead of unigrams, the performance of the model can be increased. This is important for classification problems where the word order is important. More specifically, the model obtained by running FastText with the default argument is pretty basic at classifying new sentences. So, we had to improve the performance, by changing the default hyperparameters. FastText gives the ability to autotune the model and we decided to use it with the Autotune parameter duration to 600 seconds. We also use the validation set in order to tune the parameters based on the validation set. The results that we took for the optimized model were the following: 109 dimensions, 12 epochs, 0.11474 learning rate and 1 wordNgram. Moreover, the output of the FastText are the precision and the recall. The precision is the number of correct labels among the labels predicted by FastText. The recall is the number of labels that successfully were predicted, among all the real labels.

We followed the same approach for the prediction of the structure labels but after the hyperparameter tuning we noticed that there was overfitting since the training accuracy, precision and recall were close to 1. Then, we decide to change the hyperparameters of the optimized model and in this context, we tried multiple combinations to find the one providing optimum results. Based on our trials' results we concluded that the number of epochs and the learning rate should be reduced, and the number of dimensions should be increased. The hyperparameters of the final model for the structure labels are the following: epoch=5, dimensions=110 and learning rate=0.12.

Clustering

Firstly, we should choose the appropriate number of clusters. For this reason, we decide to use the elbow method by computing the inertia value for k-means models with different predefined clusters. Also, we use a dimensionality reduction technique (PCA) in order to reduce the number of features to two dimensions and visualize the clusters.

10. Results & Quantitative Analysis (incl. visualizations)

Baseline model

	Precision	**Recall**	**F1-score**	**Support**
Claim	0.26	0.09	0.13	1197
Evidence	0.32	0.16	0.21	2173
Neither	0.74	0.91	0.81	7832
Accuracy			0.68	11202
Macro avg	0.44	0.39	0.39	11202
Weighted avg	0.60	0.68	0.62	11202

Table 1: Accuracy precision, recall and fi-score for the trainings set with argument labels

Based on the above results, we can observe the baseline model has 68% accuracy which means that the model can predict correctly the 68% of the observations. Also, we can observe that the neither label has the higher f1-score as expected since our algorithm was classified sentence as claim or evidence based on a small amount of words. If we compare the f1-score for the claim and evidence, we will observe that the algorithm makes better predictions for the evidence and it doesn't perform well regarding the claim.

FastText model

Training Argument				
	Precision	**Recall**	**F1-score**	**Support**
Claim	0.34	0.71	0.46	1193
Evidence	0.61	0.76	0.76	3590
Neither	0.95	0.84	0.89	18339
Accuracy			0.82	23122
Macro avg	0.63	0.77	0.67	23122
Weighted avg	0.86	0.82	0.83	23122
Testing Argument				
	Precision	**Recall**	**F1-score**	**Support**
Claim	0.27	0.54	0.36	253
Evidence	0.61	0.65	0.57	737
Neither	0.95	0.80	0.85	3811
Accuracy			0.77	4801
Macro avg	0.56	0.67	0.60	4801
Weighted avg	0.82	0.77	0.79	4801

Table 2: Accuracy precision, recall and fi-score for the training and testing sets with Argument labels

Based on the above results, we can observe the FastText model has 82% accuracy on the training set and 77% accuracy on the test set which means that the model can predict correctly the 77% of the observations. Also, we can observe that the neither label has the higher f1-score similarly to the baseline model. If we compare the f1-score for the claim and evidence, we will have a similar result with the baseline since f1 score from evidence is higher than the claim. However, it should be noted that both for precision and recall all the labels are higher than the baseline model.

The following tables are related with the prediction of the structure labels:

Training Structure				
	Precision	**Recall**	**F1-score**	**Support**
Background	0.81	0.63	0.71	1986
Conclusion	0.13	0.66	0.22	176
Method	0.47	0.68	0.55	792
Neither	0.88	0.80	0.84	806
Objective	0.71	0.74	0.72	1287
Result	0.85	0.65	0.74	2573
Accuracy			0.68	7620
Macro avg	0.64	0.69	0.63	7620
Weighted avg	0.76	0.68	0.71	7620
Testing Structure				
	Precision	**Recall**	**F1-score**	**Support**
Background	0.74	0.56	0.64	420
Conclusion	0.11	0.68	0.19	31
Method	0.41	0.60	0.49	165
Neither	0.70	0.68	0.69	155
Objective	0.62	0.62	0.62	277
Result	0.77	0.58	0.66	535
Accuracy			0.60	1583
Macro avg	0.56	0.62	0.55	1583
Weighted avg	0.68	0.60	0.62	1583

Table 3: Accuracy precision, recall and fi-score for the training and testing sets with Structure labels

Based on the above results, we can observe that the FastText model has 68% accuracy on the training set and 60% accuracy on the test set which means that the model can predict correctly the 60% of the observations. Also, we can observe that the neither label has the higher f1-score in comparison to the other labels. Also, it should be noted that the FastText model is unable to predict well the conclusion label since it has the lower F1-score with the precision metric to be quite low.

Clustering

The first clustering implementation that we followed was based on the document embeddings.

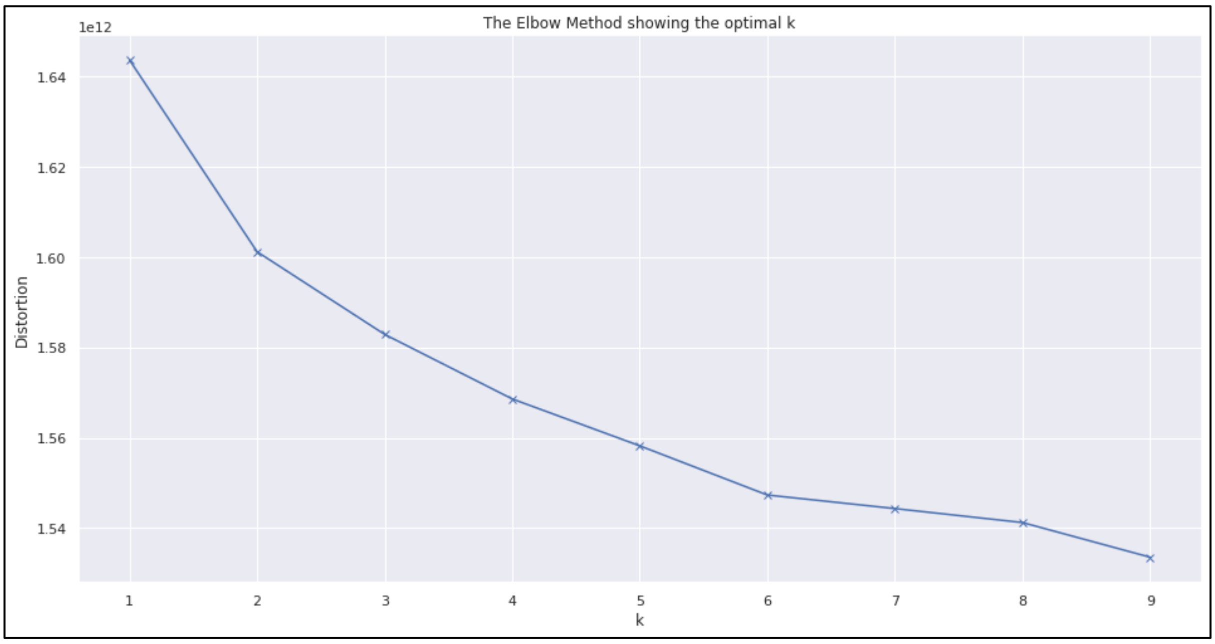


Figure 3: Elbow method to choose the number of clusters using document embeddings

Based on the elbow method from the chart above, we decided that the number of clusters that we will insert to K-Means is 6. Below, we can observe a visualization of the 6 classes in the 2-dimensonal space using PCA. We can observe that clusters 1 and 2 are less dense in comparison to the rest of the clusters.

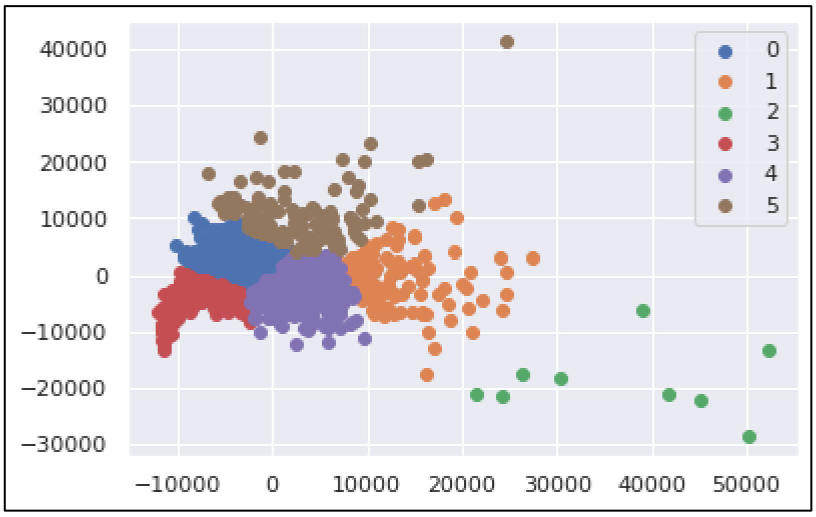


Figure 4: Visualization of the clusters from document embeddings using PCA

The following clustering implementation was based on the document embeddings and the project objective of the abstracts.

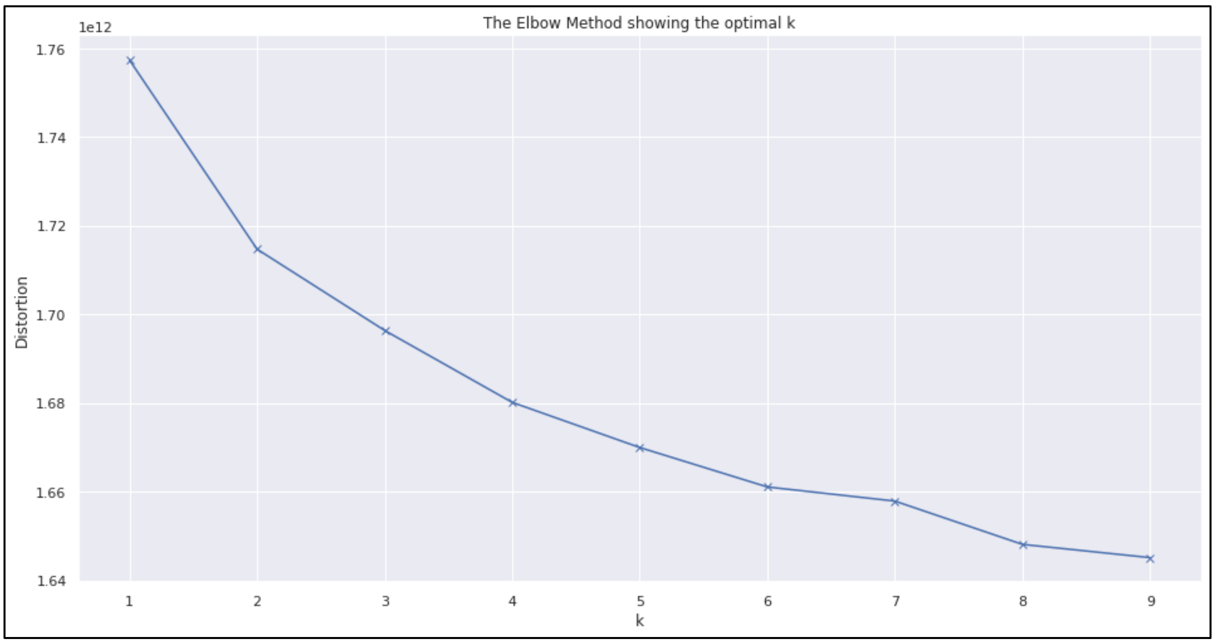


Figure 5: Elbow method to choose the number of clusters using document embeddings and project objective

Based on the elbow method from the chart above, we can observe that the starting point of the 'elbow' is the 6 clusters. As a result, the number of clusters that we will insert to K-Means is 6. Below, we can observe a visualization of the 6 classes in the 2-dimensonal space using PCA. Moreover, we can see that cluster 5 is significantly smaller than the others and it is consisted from outliers. Also, clusters 2 and 4 has smaller number of observations and the contain some outliers.

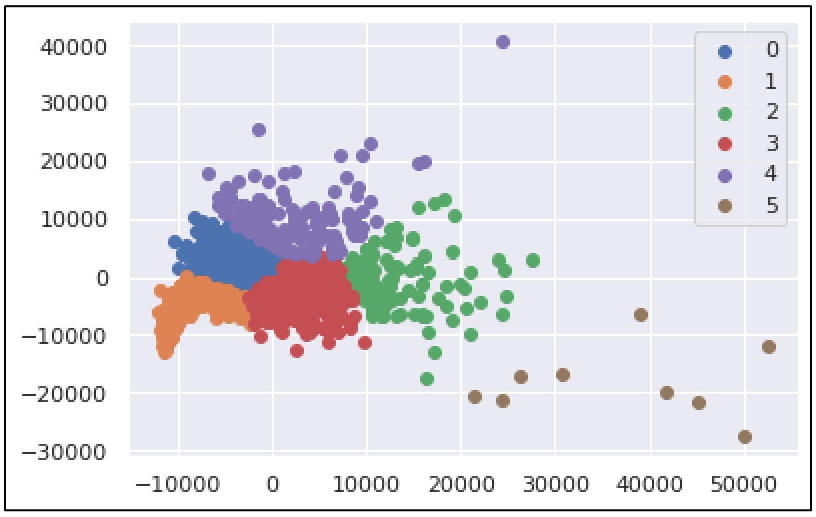


Figure 6: Visualization of the clusters from document and project objective embeddings using PCA

The final clustering implementation was based on the document embeddings and claim embeddings from the sentences.

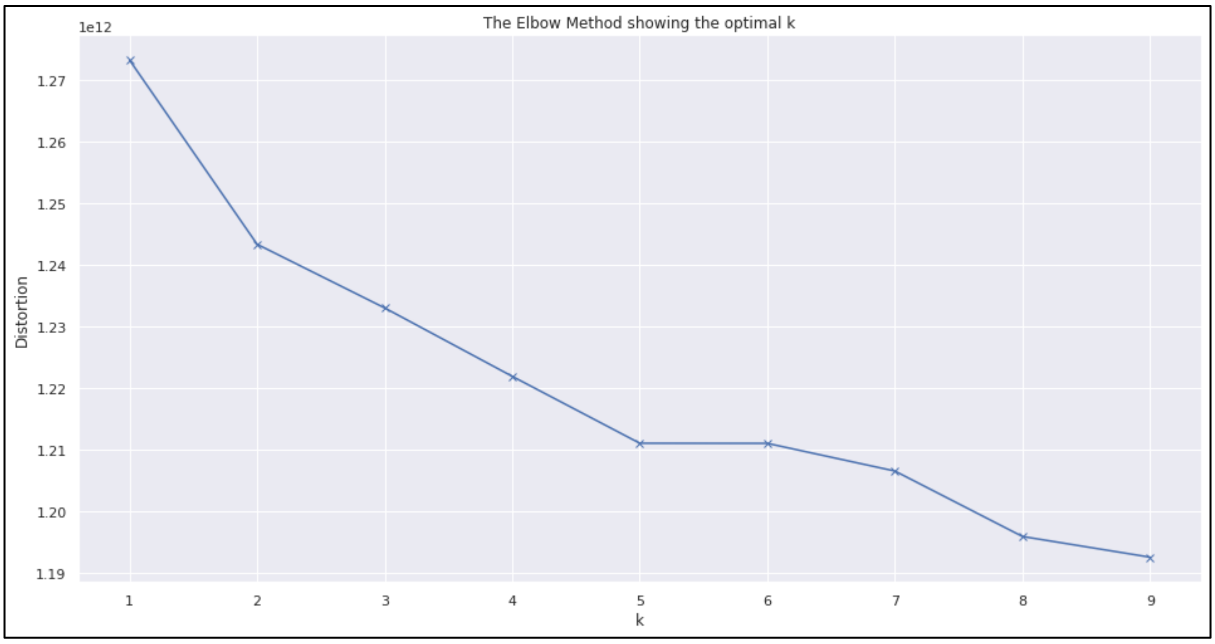


Figure 7: Elbow method to choose the number of clusters using document embeddings and claim embeddings

We can observe from the elbow chart above that the starting point of the 'elbow' is the 5 clusters. As a result, the number of clusters that we will insert to K-Means is 5. Below, we can observe a visualization of the 5 clusters in the 2-dimensonal space using PCA. We can see that cluster 4 has a significantly smaller number of observations and it is consisted mostly from outliers. Also, clusters 2 and 4 seem to have smaller number of observations and they contain some outliers.

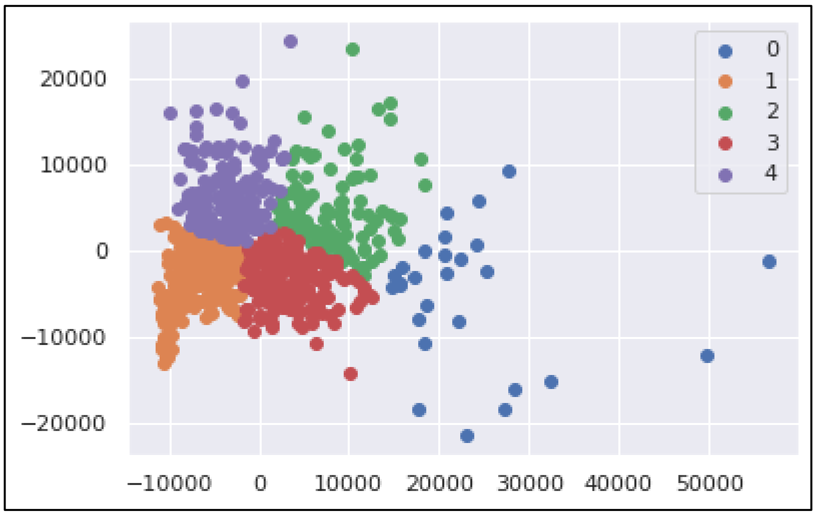


Figure 8: Visualization of the clusters from document and claim embeddings using PCA

11. Qualitative & Error Analysis

Baseline model

Based on the results from Table 1, we can say that the baseline model is quite good on predicting the neither label. However, it doesn't perform well for the other two labels which are rarer. To be more specific, the model has 0.26 precision and 0.09 recall as far as the claim label which means that from those predictions that the model classified as claims only the 26% were actually claims. Also, for all the claim sentences, the model was correctly identified only the 9% of them. For the evidence label, the numbers were better but they were still low since from those predictions that the model classified as evidence only the 32% were actually evidence and for all the evidence sentences, the model was correctly identified only the 16%. Consequently, the baseline model can perform quite well for the prediction of the majority label (Neither) but it struggles to predict correctly the other two labels and especially the claim.

FastText model

Based on the results from Table 2, we can conclude that the FastText model is doing better than the baseline model since it outperforms the baseline on every metric. However, there are areas that the FastText model predicts effectively and others which it faces difficulties. From the testing results from Table 2, we can observe that the model has 0.34 precision and 0.71 recall as far as the claim label which means that from those predictions that the model classified as claims only the 34% were actually claims. Also, for all the claim sentences, the model was correctly identified only the 71% of them. For the evidence label, the numbers were better but they were still low since from those predictions that the model classified as evidence only the 0.61% were actually evidence and for all the evidence sentences, the model was correctly identified only the 0.76%. We can observe that generally the model is quite better than the baseline model but it faces some difficulties with the precision metrics. It seems that there are many false positives during the prediction of the labels and especially for the claim labels.

Based on the results from Table 3 which is related with the structure labels, we observe that there are areas that the FastText model predicts effectively and others which it faces difficulties. From the testing results from Table 3, we can observe that the model has 0.11 precision and 0.68 recall as far as the conclusion label which means that from those predictions that the model classified as conclusions only the 0.11% were actually conclusions. Also, for all the conclusions sentences, the model was correctly identified only the 68% of them. If we observe the metrics from the other labels, we will see that the predictions were better and the results were quite close. We can say that generally the model has some good results but if fails to identify properly the conclusions.

Clustering

Firstly, we tried to create clusters by using only the documents embeddings. We used the elbow method to create the appropriate number of clusters and based on the figure 3, we decided that we should choose 6 clusters. Next, we create a clustering model using k-means and we visualized the clusters in the 2-dimensional space. Based on the figure 4, we can see that there is not perfect separation between the clusters and there're some overlapping observations.

Next, we used the elbow method for the document and project objective embedding which didn't affect the number of clusters since the elbow method showed that we should proceed with 6 clusters (figure5). The next step was to create a clustering model using k-means and we visualized the clusters in the 2-dimensional space. We compare the figure 4 with figure 6 and we didn't find a significant difference between the clusters from document and project objective embedding and clusters with document embeddings.

Finally, we combine the document and claim embeddings and we find based on the elbow method that we should proceed with 5 clusters (figure 7). Based on the figure 8, we can observe again 4 closely related clusters and a new cluster which is separated from the others.

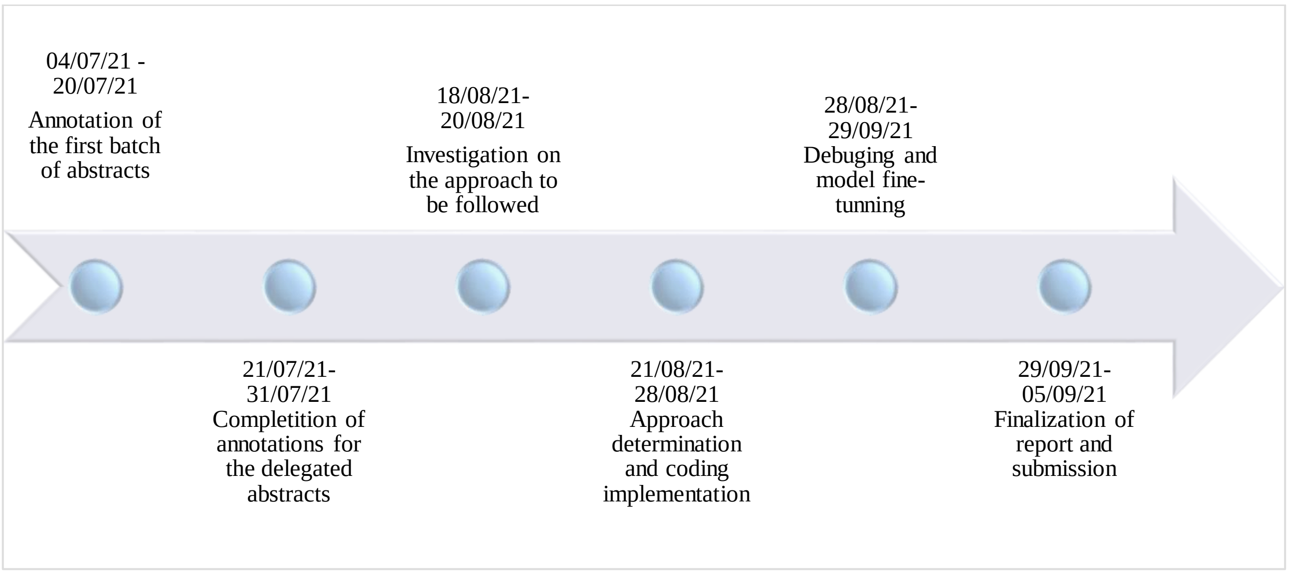
12. Discussion, Comments/Notes and Future Work

Considering that the labeling annotation of abstracts is an objective action highly dependable to the reader's experience on the topic, the data quality would be increased in case that the annotators are experts or more familiar with the abstracts' topics. Therefore, our low data quality would be considerably increased which would subsequentially increase the learning and predictive ability of our model. Additionally, our work could be improved and the prediction results of our model could be more efficient with the use of additional pre-trained FastText models or by applying CNN unimodals or mutlimodal models with shared embedding layer or RNN models that would exhibit temporal dynamic behavior.

13. Members/Roles

Name	Role
Ioanna Kanellou	ML Engineer Data Analyst
Georgios Mavridis	ML Engineer Data Analyst
Evangelia Papantoni	ML Engineer Data Analyst

14. Time Plan


04/07/21 - 20/07/21
Annotation of the first batch of abstracts
21/07/21- 31/07/21
Completition of annotations for the delegated abstracts
18/08/21- 20/08/21
Investigation on the approach to be followed
21/08/21- 28/08/21
Approach determination and coding implementation
28/08/21- 29/09/21
Debuging and model fine-tunning
29/09/21- 05/09/21
Finalization of report and submission

15. References

- https://spiral.imperial.ac.uk/bitstream/10044/1/44024/4/FAIA287-0219.pdf
- https://direct.mit.edu/coli/article/45/4/765/93362/Argument-Mining-A-Survey
- https://FastText.cc/docs/en/supervised-tutorial.html
- https://towardsdatascience.com/FastText-bag-of-tricks-for-efficient-text-classification-513ba9e302e7
- https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

- https://towardsdatascience.com/FastText-sentiment-analysis-for-tweets-a-straightforward-guide-9a8c070449a2

16. Appendices

The source code where all figures, tables and results are presented is available to GitHub.