Machine Learning and Content Analytics

Pilot Project - Argumentation Mining

Professor:

Haris Papageorgiou

Students:

Nikolaos Kyriakakis (f2822006)

Georgios Papadomarkakis (f2822021)

Contents

Table of Figures

1.1 Introduction

Argumentation Mining became a hot topic, attracting the interests of several and diverse research communities, ranging from artificial intelligence to computational linguistics, natural language processing, social and philosophical sciences.

Argumentation mining is an advanced form of human language understanding by the machine. This is a challenging task for a machine. When sufficient explicit discourse markers are present in the language utterances, the argumentation can be interpreted by the machine with an acceptable degree of accuracy. However, in many real settings, the mining task is difficult due to the lack or ambiguity of the discourse markers, and the fact that a substantial amount of knowledge needed for the correct recognition of the argumentation, its composing elements and their relationships is not explicitly present in the text but makes up the background knowledge that humans possess when interpreting language.

In this project, we attempt to describe the problems and challenges of argumentation mining from a machine learning angle. This is achieved by firstly collecting publications that are related to the medical science sector. These publications were annotated into argumentation statements (Claim, Evidence) as also into a structure format (Background, Objective/Aim, Methods, Results, Conclusions). After the annotation process was completed by all teams participating in the project three different datasets were formed, one with the argumentation labels, the other with the structure labels and the final one that was about the citations of the abstracts. Finally, these datasets were used as an input for abstracts classification and clustering by using machine learning techniques.

The business perspective of this project is about supporting sense-making, practical reasoning and writing support of scientific papers.

1.2 Our project

Annotation

Argumentation is an intelligent communication task that is inherent to human behavior. It is a discourse activity that aims at increasing or decreasing the acceptability of a controversial claim or point of view. Humans are very keen to convince others of their opinion and bring arguments to a discussion to support their claims.

Argumentation is a verbal, social, and rational activity aimed at convincing a reasonable critic of the acceptability of a standpoint by putting forward a constellation of propositions justifying or refuting the proposition expressed in the standpoint. Let us proceed to an analysis of this above definition of argumentation.

The process of argumentation which summarizes the overall problem can be expressed with the following:

1. Identify argumentative text (or a portion of a text).
2. Segment the text into argumentative discourse units (ADUs).
3. Identify the central claim.
4. Find supporting and objecting statements.
5. Derive the structure of arguments.
6. Identify the type and the quality of the argumentation.

Finding arguments in an automated way in human discourse was early on discovered as a desirable characteristic of intelligent machines, often referred to as argumentation mining. This process has been defined as the automated detection of the argumentation structure and classification of both its component elements and their argumentative relationships.

Over the years many argumentation models that expect to find a typical argumentation structure have been defined, the most popular being the model of Toulmin [1]. He has suggested an argumentation model composed of six component types: claim, data, qualifier, warrant, rebuttal and backing, which recently has been simplified to aid automated recognition into a claim and its premises or into a claim and its premises where a distinction is made between supporting and attacking elements [2].

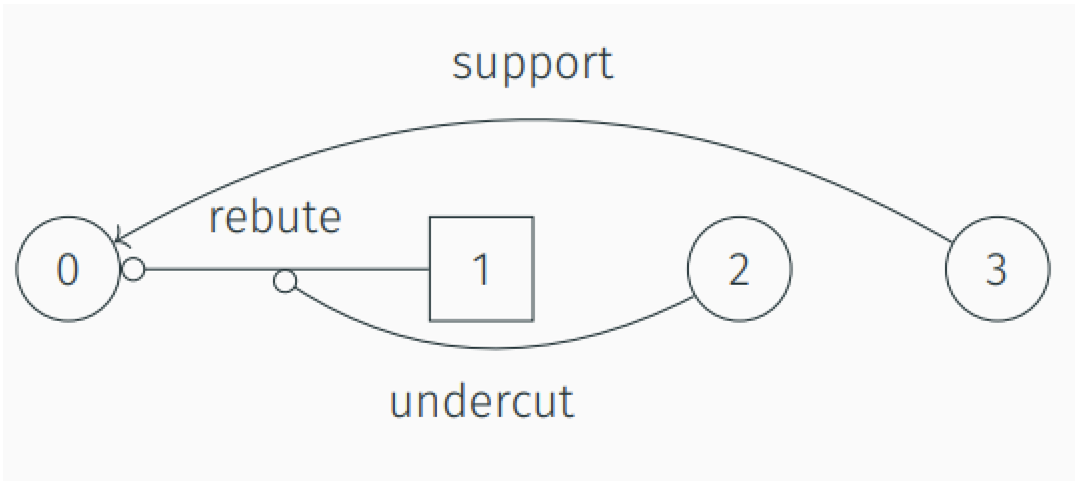


Figure 1: Graph representation of argumentation structure example.

Toulmin model of argumentation structure is presented at Figure 2 below.

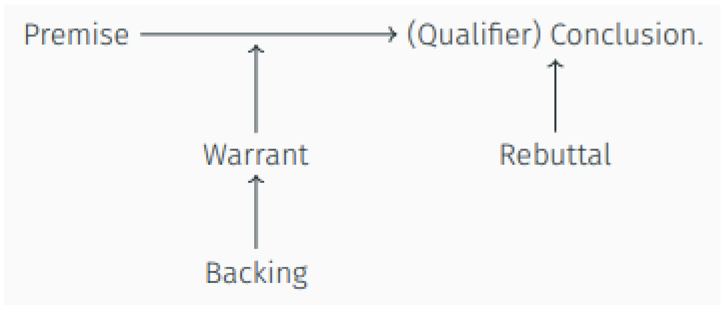


Figure 2: Toulmin model

- Claim: the statement that the speaker wants the listener or reader to accept, as much as possible.
- Premises: the data and facts offered to support the claim
- Warrant: connects the grounds or justifications to the claim, it mainly justifies the support relation or "inference" from the grounds to the claim.
- Backing: supports and explains the warrant, it provides trustworthiness to the warrant.
- Qualifiers: these are statements about the strength of the claim.
- Rebuttals: these are exceptions to the claim. Rebuttals develop situations in which a claim may be defeated.

In this project the two argumentative statements we annotated were the Claim and the Evidence.

- Claim: an argumentative statement that reports the study findings and derives from the author's original work.
- Evidence: the data (observational or measurements) and facts offered to support the Claim.

Furthermore, the abstracts were annotated on their structure format (Background, Objective/Aim, Methods, Results, Conclusions). Finally, publications that cited to the abstracts that were annotated before were found. These citations references were annotated as if they cite a claim and if the citation is Positive or Negative. This whole annotation process produced three different datasets which were later used as input for the argumentation mining project and more specifically in the classification and clustering techniques.

Argumentation Mining

Recognizing argumentation structures in text is considered a difficult task both for humans and machines. Arguments are sometimes revealed by the presence of discourse markers (e.g., in English adverbs such as "because"), but such markers have often ambiguous meanings or are missing. Moreover, the premises in favor of the same claim might be far apart in a discourse, making it difficult to automatically link them together. Also, argumentation structures might take the form of graphs of nested tree structures.

The goal of argumentation mining is to automatically extract arguments and their relations from a given document. The majority of argumentation mining systems follows a pipeline scheme, starting with simpler tasks, such as argument component detection, down to more complex tasks, such as argumentation structure prediction.

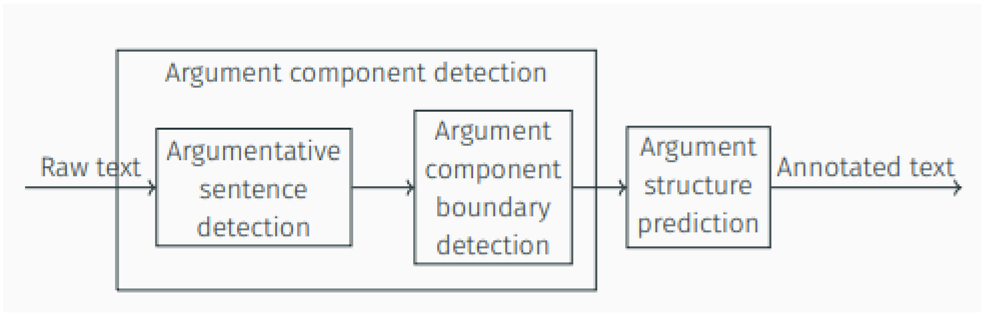

Argument component detection
Raw text
Argumentative sentence detection
Argument component boundary detection
Argument structure prediction
Annotated text

Figure 3: Pipeline architecture of an AM system.

Classification of arguments

Argumentative sentence detection addresses the task of extracting those sentences in the input document that contain an argument (or part of it), and that can therefore be defined as argumentative. In general, we have three options:

1. a binary classifier is trained to distinguish argumentative from non-argumentative sentences.
2. a multi-class classifier is trained to discriminate all the argument components that exist in the adopted argument model.
3. a set of binary classifiers is trained, so that a sentence can be predicted to contain more than one argument component.

I our project a multi-class classifier will be used in order to classify sentences into three categories, Claim, Evidence and Neither.

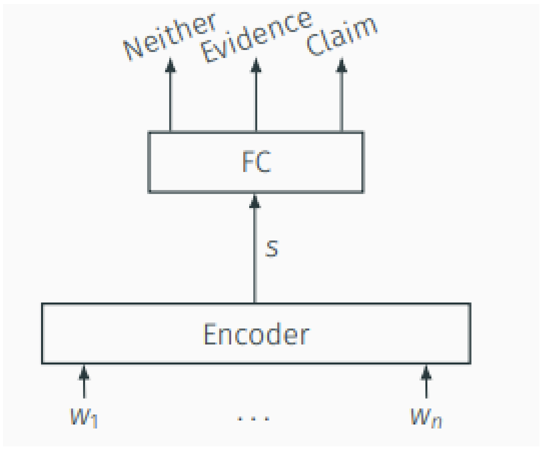


Figure 4: A model that can classify sentences as argumentative /non-argumentative or to one of the available categories.

We used this multi-class classifier which was based on a machine learning algorithm of Facebook named Fasttext. This algorithm is specifically used for text processing and classification. Finally, the same algorithm was used with the dataset which contained the structure labels as input. Statistics measures such as accuracy, precision, recall and f1 score was measured for all our models (classifiers) and were compared to a basic classification algorithm we formed and used as an intuitive baseline model.

Clustering of arguments between different abstracts

The final stage aims at predicting links between arguments, depending on the target granularity. This represents an extremely challenging task, as it requires to understand connections and relationships between the detected arguments, thus involving high-level knowledge representation and reasoning issues. The problem's goal is usually referred to as prediction rather than detection, since the target is not just a specific portion, but rather a connection (or link) between arguments of different abstracts. The output of this stage is a graph connecting the retrieved arguments.

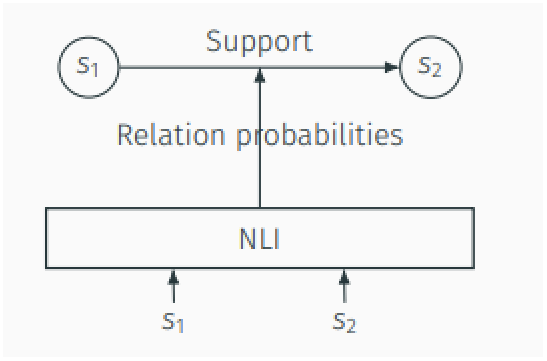


Figure 5: A model to calculate the probabilities and the type of a relation

In this project we tried to output two different graphs after extracting the sentence embeddings from the machine learning models. In the first graph edges were formed which connect claim only arguments from different abstracts. In the second graph edges were formed which connect claim and evidence arguments in the abstracts. Finally, we focused on finding the communities and cliques of different scaling in order to form our clusters and later check our results with quantitative analysis.

1.3 Our Vision/Goals

Since data in this project refer to articles about medical science, we present three goals:

- ✓ Support sense-making:

Sense-making refers to making sense of larger or disparate amounts of information to construct narratives or explanations, as an individual or as a group. Therefore, a good classification or clustering model could be used by people in the medical sector, when they need to further support their claim or a decision they need to take. By making use of the clustering model someone can find related articles to the one he is interested in and therefore further extend the amount of information on this specific subject.

- ✓ Practical reasoning:

Practical reasoning refers to deciding on a practical course of action. Our models could be used by doctors, pharmacists or people in the medical industry to choose the best treatment or medication for a case by gaining extended knowledge on a specific disease.

- ✓ Argument retrieval:

Retrieving arguments from text collections. These arguments can be used to provide further feedback on each subject. This extra information can be used for someone who is interested in this subject either for future work on writing a scientific article or support their or other decisions.

2.1 Methodology

The execution of Argumentation Mining Project begun with a series of e-Labs in which, teams were given significant guidance and instructions for the course of the project by the team leader, Aris Fergadis. To begin with, the first step of the whole procedure was to get familiar with a set of rules and theories concerning argumentation, annotation components, citances and abstract structure relevant to a dataset of scientific abstracts linked with the Sustainable Development Goals of the United Nations.

Argumentation is a complex phenomenon which throughout the procedure of argument mining we target to extract prototypical arguments or part of them with a reasonably good accuracy. Argument mining depends on the goal that is pursued which differs. The objective can be:

- the finding of argumentative statements, which in our case was the goal
- the finding of supporting and objecting statements
- derive the structure of arguments
- assessing argumentation

The finding of argumentative statements is related with the claims in the abstracts, sometimes followed by the relevant evidence which the author uses in order to support the claim. So, the two argumentative statements we annotate for are:

- claim, which reports the study findings
- evidence, which presents facts to support the claim

Our task at first level was to identify and annotate the argumentative statements for each abstract specifying whether claim or evidence exist in each sentence. The existence of each argumentative statement occurred based on a theoretical set of rules. Those rules could be phrases or words which indicate the existence of a claim or evidence. Also, the nature of each sentence without any specific word or phrase could also indicate whether a sentence could be a claim or evidence.

To continue, another part of the annotation phase was relevant to citances. Citances are the sentences that refer to the work of other authors. For the papers we study our purpose was to identify whether a citance was:

- Positive – Supporting: the citation shows that the author's base verifies the claim or the findings of the abstract
- Negative – Contrasting: the citation states that the authors question / argue the claim or the findings of the abstract
- Neutral – Mentioning: the citation mentions the reference paper without a standpoint
- Irrelevant: the citation seems to have nothing in common with the reference article

Furthermore, the last part of annotation is related with the structure of the abstract which most authors tend to follow. This structure differs according to each abstract's domain, but we can argue that there is a basic rule which declares five categories of sentences that the majority of scientific abstracts follow. These are :

- Introduction : This is the background of the problem and what is the problem
- Objective / Aim: This is the sentences or sentences on which the aim of the abstract is referred
- Method: These sentences answer the question how a particular abstract approached the issue
- Results: These sentences answer the question what the author found
- Conclusion: What do the abstract's findings mean.

Collectively, a sequence of steps was created which every team had to follow in order to for the annotation procedure to be implemented. Briefly, these steps:

1. Focus on the title
2. Identify the structure of the abstract
3. Focus on the objective / aim
4. Identify the claim
5. Identify the evidence
6. Categorize the citances

Based on those rules and theories, at first level, team leader distributed a set of ten abstracts to every team. The aim was for every team to annotate (each member separately) all the above categories and pursue a satisfying threshold of agreement rate. So, after getting familiar with the annotation procedure another portion of a hundred and five abstracts were given to each team to annotate.

Afterall, after the problematic abstracts have either been corrected or deleted and given the desired agreement rates the final dataset was formed. To this point, it is mandatory to state that the label of each sentence, for every annotated category was implemented with MACE's function prediction which assigns a sufficient rate of agreement to each annotator. MACE (Multi Annotator Competence Estimation) is used when evaluating redundant annotations and we usually want to:

- Aggregate annotations to recover the most likely answer
- Find out which annotators are trustworthy
- Evaluate item task and difficulty

MACE solves all these kind of problems by learning competence estimates for each annotator and computing the most likely answer based on competencies.

After the formation of all datasets was completed, we proceed to the next phase of the project, the Argumentation Mining. In this stage there are three steps to be implemented which are the:

- Intuitive Baseline
- Fasttext Approach
- Abstract Clustering using Graph

The objective of the Intuitive Baseline was applied only for the argument labels and its approach was to create a simple script which would classify each sentence as claim, evidence or neither. It was based on exploratory data analysis which created simple empirical rules so when certain word occurred each sentence was classified accordingly. For example, when the word claim existed in a sentence then, that sentence was classified as claim. The purpose of the Intuitive Baseline approach was to conclude to a confusion matrix to obtain an initial accuracy, precision and recall of a simple mining approach so, later on, to be comparable with the results of more advanced machine learning techniques.

In addition, the second step included a Fasttext model implementation approach for both argument and structure labels which was divided into two versions. The objective of the first approach was to:

- transform the text into the appropriate form as input for the model
- train a supervised fasttext model
- implement the hyper-parameter tunning to conclude to the best model
- conclude to a confusion matrix in order to make comparisons and evaluation

Moreover, the second approach included the extraction of context vectors, either abstract or sentence, from a pre-trained fasttext model in order to feed an MLP model and then compare its results to those of version one.

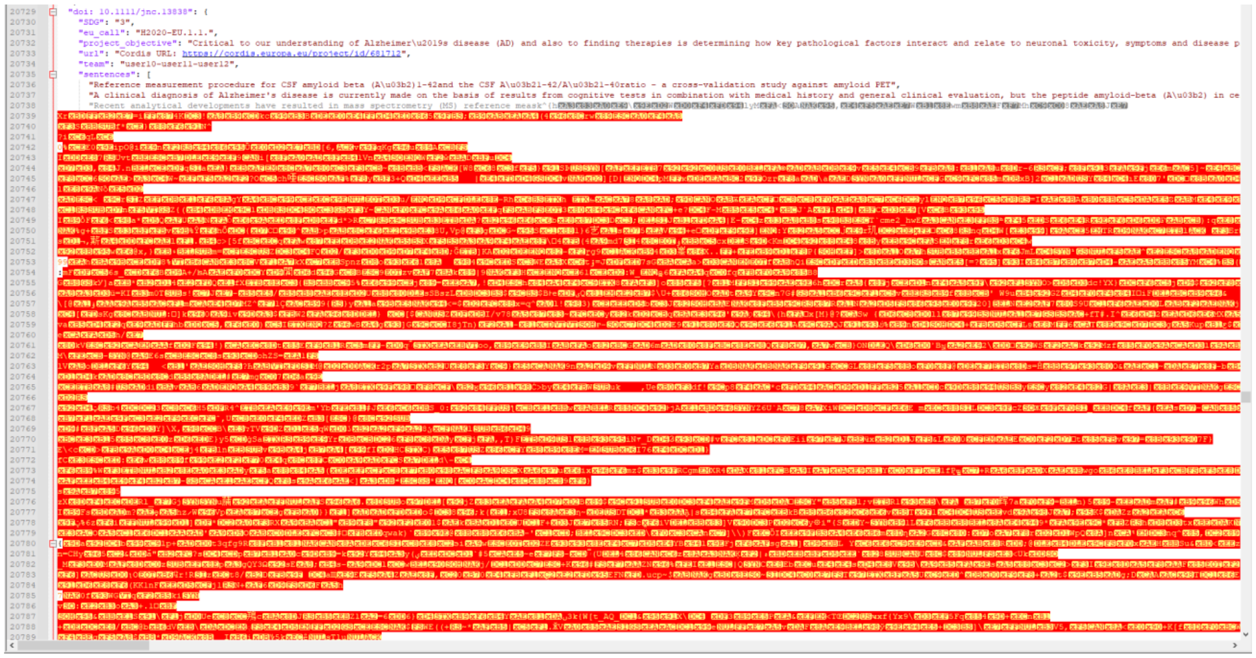
Finally, the last step of Augmentation Mining Project is related with abstracts' clustering using Graph method of NetworkX library in python. In that case, the procedure begins with extracting argument embeddings at sentence level and creating a graph in which each abstract id is assigned to each node. Then, based on the similarity of the extracted embeddings edges were created between the most similar nodes and in that way the graph obtained its initial form. After experimenting with edges' weights and for various values of k-closest arguments we are able to make significant inferences regarding graph's communities, cliques as well as for the closest or more similar abstracts.

2.2 Data Collection

Our initial data are scientific articles which were created following a call from the European Union which announces projects which are to be funded. These abstracts have originally been split by the project leader into sentences and have been distributed to the teams of the project in order to become annotated.

Following the annotation procedures described above, each team tried to reach satisfactory agreement rates for each sentence label. At this point we must emphasize that a sentence can be characterized by both argumentative and structure labels. Achieving the desired agreement percentages using the MACE function was a difficult process as sometimes the label assigned to a sentence depends on the subjectivity of each member of a team. In many cases, in fact, some abstracts had to be deleted as they were either in bad shape or did not make sense or good agreement rates could not be reached in them.

After each team reached the point of satisfying the desired agreement rates, then every sentence was characterized with a label relative to type of the dataset it belongs to. In addition, after the final configuration of the data by the project leader, in particular in dataset 'dataset_aueb_structure_v2' a specific abstract was deleted as it was of unknown format as we can see in the image below. This step was also the last to create the final datasets.

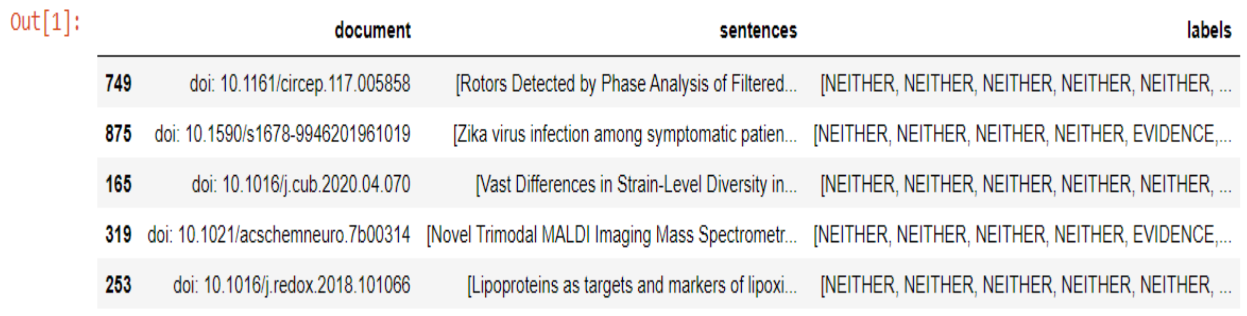


```
"doi: 10.1111/jnc.13838": {
  "SDG": "3",
  "eu_call": "H2020-EU.1.1.",
  "project_objective": "Critical to our understanding of Alzheimer\u2019s disease (AD) and also to finding therapies is determining how key pathological factors interact and relate to neuronal toxicity, symptoms and disease p
  "url": "Cordis URL: https://cordis.europa.eu/project/id/681712",
  "team": "user10-user11-user12",
  "sentences": [
    "Reference measurement procedure for CSF amyloid beta (A\u03b2)1-42and the CSF A\u03b21-42/A\u03b21-40ratio - a cross-validation study against amyloid PET",
    "A clinical diagnosis of Alzheimer's disease is currently made on the basis of results from cognitive tests in combination with medical history and general clinical evaluation, but the peptide amyloid-beta (A\u03b2) in ce
    "Recent analytical developments have resulted in mass spectrometry (MS) reference meask^(h[illegible]
[illegible]
```

2.3 Dataset Overview

After the desired agreement threshold has been reached and problematic abstracts no more existed in the whole dataset, then, we were able to have an initial dataset overview. There were three datasets created and given to each team.

The first two datasets were related to argument procedures, were named 'dataset.json' and dataset_aueb_argument_v3.json respectively and in every case were concatenated as they were used for same purposes. Their form is depicted in figure 6.



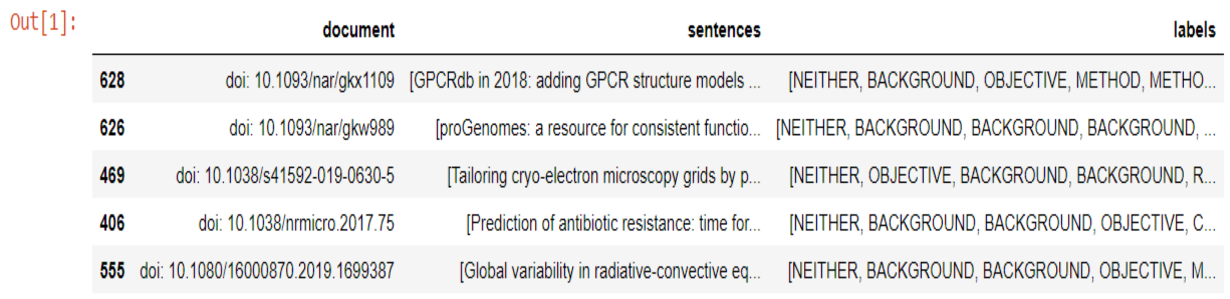
Out[1]:

	document	sentences	labels
749	doi: 10.1161/circep.117.005858	[Rotors Detected by Phase Analysis of Filtered...	[NEITHER, NEITHER, NEITHER, NEITHER, NEITHER, ...
875	doi: 10.1590/s1678-9946201961019	[Zika virus infection among symptomatic patien...	[NEITHER, NEITHER, NEITHER, NEITHER, EVIDENCE,...
165	doi: 10.1016/j.cub.2020.04.070	[Vast Differences in Strain-Level Diversity in...	[NEITHER, NEITHER, NEITHER, NEITHER, NEITHER, ...
319	doi: 10.1021/acschemneuro.7b00314	[Novel Trimodal MALDI Imaging Mass Spectrometr...	[NEITHER, NEITHER, NEITHER, NEITHER, EVIDENCE,...
253	doi: 10.1016/j.redox.2018.101066	[Lipoproteins as targets and markers of lipoxi...	[NEITHER, NEITHER, NEITHER, NEITHER, NEITHER, ...

Figure 6: Arguments' dataset overview

As we can see, the first column contains the abstract's id, the second one contains the abstracts sentences and the third one contains a list with the labels which came up and assigned from all teams' agreement rate. It goes without saying that the column sentences also contain a list of sentences, and both lists of columns 'sentences' and 'labels' have the same length as each sentence matches to a label.

Additionally, the ‘dataset_aueb_structure_v2’ was created and its formation is similar to the previous one as figure 7 shows. The first column contains the abstract’s id while the two other columns are lists of same length which contain the sentences of the abstract and the labels of structure respectively.



Out[1]:

	document	sentences	labels
628	doi: 10.1093/nar/gkx1109	[GPCRdb in 2018: adding GPCR structure models ...	[NEITHER, BACKGROUND, OBJECTIVE, METHOD, METHO...
626	doi: 10.1093/nar/gkw989	[proGenomes: a resource for consistent functio...	[NEITHER, BACKGROUND, BACKGROUND, BACKGROUND, ...
469	doi: 10.1038/s41592-019-0630-5	[Tailoring cryo-electron microscopy grids by p...	[NEITHER, OBJECTIVE, BACKGROUND, BACKGROUND, R...
406	doi: 10.1038/nrmicro.2017.75	[Prediction of antibiotic resistance: time for...	[NEITHER, BACKGROUND, BACKGROUND, OBJECTIVE, C...
555	doi: 10.1080/16000870.2019.1699387	[Global variability in radiative-convective eq...	[NEITHER, BACKGROUND, BACKGROUND, OBJECTIVE, M...

Figure 7: Structure's dataset overview

The datasets for the arguments contain in the ‘labels’ column the values: ‘CLAIM’, ‘NEITHER’ and ‘EVIDENCE’ while the structure dataset contains the labels: ‘NEITHER’, ‘BACKGROUND’, ‘OBJECTIVE’, ‘RESULTS’ and ‘CONCLUSION’ as well.

2.4 Data Processing/Annotation/Normalization

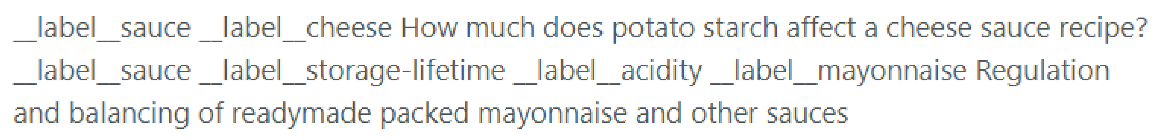
Once the datasets to be used have been created, the next step is to pre-process and modify them properly so that they can be used by the models in each step. As already mentioned, the project consists of three basic algorithmic steps which have some common data processing methods but also each one has its own peculiarities.

In order to obtain remarkable results so that there can be a comparison between methods on the exact same data in each of the three procedures the data were separated by a specific random_state parameter in train, validation and test dataset. As a team, we decided to keep 60% of the instances for the train dataset and another 20% for the validation and the test dataset. It was important for each step to include every instance to the corresponding dataset in order for metrics referring to same abstracts to be created.

As for the Intuitive Baseline part of the project, no other data processing or transformation process took place. Text processing in this phase did not serve any purpose as we did not want words to be excluded or to reform as the empirical rules would no longer be possible to be identified. So, before running the script both columns of abstract’s sentences and labels were unlisted in order for each one to match with a label.

On the contrary, in the Fasttext approach the pre-process of the text was necessary and inevitable so for the model to learn the more essential part of all abstracts. Initially, after all the abstracts were joined together to form a single dictionary, we created the list of the most common words. From this list we chose to remove the hundred most frequently displayed. Then, having introduced the nltk library, we downloaded from it a ready list of stop words so that they can be removed from the text as well. Finally, the last technique between NLP pre-processing techniques which was used was the transformation of all texts into lowercase text.

Then, after the abstracts were split again in their individual sentences, as well as the lists with the corresponding labels, it was necessary to proceed with an additional editing of the text so that it could be given as input to the Fasttext model. In order to train and evaluate the Fasttext classifier the appropriate format which Fasttext expects as figure 8 depicts requires the prefix '__label__' before each sentence's label. Lastly, after labels were accompanied by the required prefix, columns of sentences and labels were merged in order for the procedure of text's processing to be completed.



```
__label__sauce __label__cheese How much does potato starch affect a cheese sauce recipe?
__label__sauce __label__storage-lifetime __label__acidity __label__mayonnaise Regulation
and balancing of readymade packed mayonnaise and other sauces
```

Figure 8: Fasttext's input required format

Proceeding to the final stage of the project regarding the clustering of abstracts through writing, the required transformations and procedures concerned the conversion of text into embeddings and then the creation of the table of distances between them. The text was converted to numerical representation (embeddings) with the SentenceTransformer function from SentenceTransformer library. Having now each sentence with a numerical representation, the aim is to form a table of distances based on simple statistical procedures which are described below:

- The initial table with the numerical representations of the embeddings is normalized
- Using the pairwise_distances function from the sklearn library we create by inputting the normalized table an initial distance table between all embeddings
- The new table of posts is normalized and then converted to a table of similarities by removing all the elements from the unit
- The table is transformed into an upper triangle as it is symmetrical, and we want to avoid multiple and unnecessary access to it

Then having as a given the upper triangular table of distances, considering a minimum limit of similarity we record the indexes between the similar abstracts which correspond to their doc id. After clearing the same doc ids, we can now form a data frame with columns: 'from', 'to' and with the corresponding similarities as weights the base of our desired graph has already been created.

2.5 Algorithms, NLP architectures/systems

Natural language processing (NLP) is an area of computer science and artificial intelligence concerned with the interaction between computers and humans in natural language. NLP is an interdisciplinary field concerned with the interactions between computers and natural human languages (*e.g.*, English) — speech or text. NLP powered software helps us in our daily lives in various ways, for example:[3]

• Personal assistants: Siri, Cortana, and Google Assistant.

• Auto complete: In search engines (e.g., Google).

• Spell checking: Almost everywhere, in your browser, your IDE (e.g., Visual Studio), desktop apps (e.g., Microsoft Word).

• Machine Translation: Google Translate.

NLP is mainly divided into two fields: Linguistics and Computer Science.

The Linguistics side focuses on understanding the structure of language, including the following sub-fields:

Phonetics: The study of the sounds of human language.

Phonology: The study of the sound systems in human languages.

Morphology: The study of the formation and internal structure of words.

Syntax: The study of the formation and internal structure of sentences.

Semantics: The study of the meaning of sentences.

Pragmatics: The study of the way sentences with their semantic meanings are used for communicative goals.

The Computer Science side is concerned with applying linguistic knowledge by transforming it into computer programs with the help of sub-fields such as Artificial Intelligence (Machine Learning and Deep Learning).

Some of the most popular techniques of Natural Language Processing are being described below [4] :

- Named Entity Recognition (NER)
  This technique is one of the most popular and advantageous techniques in Semantic analysis, Semantics is something conveyed by the text. Under this technique, the algorithm takes a phrase or paragraph as input and identifies all the nouns or names present in that input
- Tokenization
  Tokenization, it is basically the splitting of the whole text into the list of tokens, lists can be anything such as words, sentences, characters, numbers, punctuation, etc. Tokenization

has two main advantages, one is to reduce search with a significant degree, and the second is to be effective in the use of storage space. The process of mapping sentences from character to strings and strings into words are initially the basic steps of any NLP problem because to understand any text or document we need to understand the meaning of the text by interpreting words/sentences present in the text.

- Stemming and Lemmatization
  The increasing size of data and information on the web is all-time high from the past couple of years. This huge data and information demand necessary tools and techniques to extract inferences with much ease.
  "Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form - generally a written form of the word." For example, what stemming does, basically it cuts off all the suffixes. So, after applying a step of stemming on the word "playing", it becomes "play", or like, "asked" becomes "ask". Lemmatization usually refers to do things with the proper use of vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. In simple words, Lemmatization deals with lemma of a word that involves reducing the word form after understanding the part of speech (POS) or context of the word in any document.
- Bag of Words

  Bag of words technique is used to pre-process text and to extract all the features from a text document to use in Machine Learning modelling. It is also a representation of any text that elaborates/explains the occurrence of the words within a corpus (document). It is also called "Bag" due to its mechanism, i.e., it is only concerned with whether known words occur in the document, not the location of the words.

The first approach of the project, without special NLP techniques, in order to obtain a form of knowledge from the computer is done through the Intuitive Baseline approach. As already mentioned, the result of this approach was the classification of sentences as 'CLAIM','EVIDENCE', or 'NEITHER' without actually training a classifier. The text classification was based on empirical rules from which the aim was to obtain a confusion matrix for later comparison of the results with more advanced machine learning methods.

The Fasttext approach as a more advanced technique of text classification was then applied. Text classification is a core problem to many applications, like spam detection, sentiment analysis or smart replies. In this tutorial, we describe how to build a text classifier with the Fasttext tool. Fasttext is an open-source library, developed by the Facebook AI Research lab. Its main focus is on achieving scalable solutions for the tasks of text classification and representation while processing large datasets quickly and accurately.

As mentioned previously, we need labelled data to train our supervised classifier. Each line of our dataset contains label, followed by the corresponding sentence. All the labels start by the __label__ prefix, which is how Fasttext recognize what is a label or what is a word. The model is then trained to predict the label given the sentence. After the model has been trained, we continue to the validation procedure which intends to make the model better by experimenting with various hyperparameters which, for the Fasttext approach are [5] :

- Epochs
  epoch is the number of times a model sees a phrase or an example input. By default, the model sees an example five times, i.e., epoch = 5. We can increase that to 25 using the -ecpoch option to make the model 'see' an example sentence 25 times, which can help the model in learning better
- Word n-gramms
  In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. The rang of -ngrams hyperparameter for a Fasttext model is in between [1-5]
- Learning Rate
  Learning rate of an algorithm indicates how much the model changes after each example sentence is processed. We can both increase and decrease the learning rate of an algorithm. A learning rate of 0 means that there is no change in learning, or the rate of change is just 0, so the model doesn't change at all. A usual learning rate is 0.1 to 1.

- Loss functions
  Since we are training our model on a few thousands of examples, the training only takes a few seconds. But training models on larger datasets, with more labels can start to be too slow. A potential solution to make the training faster is to use the hierarchical softmax, instead of the regular softmax. This can be done with the option -loss hs. The hierarchical softmax is a loss function that approximates the softmax with a much faster computation.

Regarding the last step of the project, clustering, our team decided to approach this issue with the Graph method. The main idea is that after converting each sentence of the abstracts into embedding (numerical representation), based on the distances that will occur (as described in the previous section) we can create a form of connection between all the argumentative statements on a scale that will be the edges of the later graph. Then, after we have experimented with the intensity of the connection that we want to have, we have the possibility through advanced tools to form our graph in a precise level and then using appropriate commands to extract information concerning the cliques, the societies but also the similar argumentative statements. The desired result will be the clustering of abstracts in similar but also at the same topics.

The whole procedure begins with the transformation of each sentence to an embedding. An embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embeddings make it easier to do machine learning on large inputs like sparse vectors representing words. Ideally, an embedding capture some of the semantics of the input by placing semantically similar inputs close together in the embedding space. An embedding can be learned and reused across models [6]. As previously mentioned in order for the embeddings to be created SentenceTransformer was utilized.

The implementation of the Graph was achieved using the NetworkX python package. NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks [8]. The use of this library is intended to create a script and easy management in order to extract information. The types of information we are interested in are cliques, communities and the most closely related abstracts through similar embeddings.

3.1 Experiments – Setup, Configuration

Intuitive Baseline Model

The first step of our project was about forming an intuitive baseline model on the dataset that contained the argument labels. The purpose of this baseline model is to produce a script which will classify the sentences into arguments (Claim, Evidence) after conducting exploratory data analysis on the dataset.

Following this exploratory data analysis of the dataset and taking into consideration the whole annotation process that we conducted on the articles before this step we formed a lexicon which will be used to either classify the sentences into Claim or Evidence.

- Lexicon used for claim arguments: [provide, reveal, confirm, suggest, claim, conclusion, overall, summary, indicate, prove, finally]

This lexicon was produced by using the empirical knowledge of the annotation process as also by following some rules. More specifically the rules which we based our lexicon and our model on were:

- Claims are usually concluding sentences. Claims should derive from the authors' research/study findings, even if there no explicit evidence in the abstract that supports those claims.
- Verbs like "reveal", "provide", "confirm", "suggest", "indicate", "prove" potentially show that a scientific claim is coming.
- This also holds for nouns like "conclusion", "claim" and adjectives like "overall", "summary" and "finally".

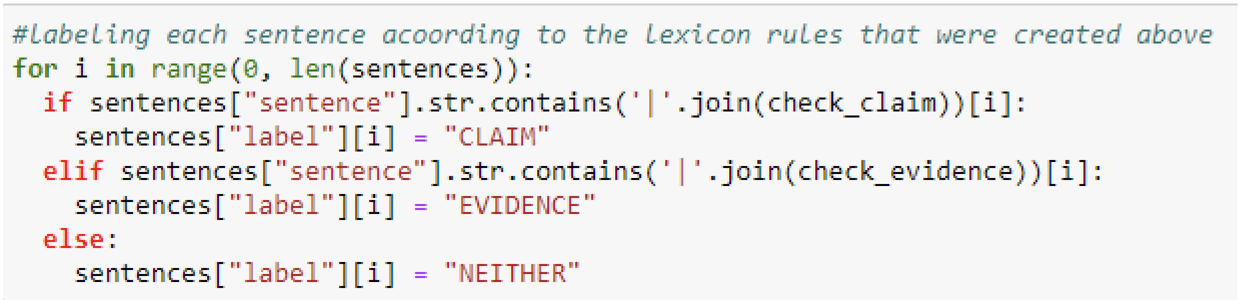
By following a similar process, we created a lexicon in order to characterize sentences as Evidence.

- Lexicon used for evidence arguments: [evidence, result, finding, %]

The rules based on which the lexicon and later on the whole model was based on are:

- ➢ Evidence is usually found in sentences that report experimental results or observations. Therefore, every sentence that contained a percentage was marked as evidence by searching for the "%" symbol.
- ➢ Evidence is usually found in sentences just before the sentence that is characterized as a claim argument.
- ➢ Nouns like "evidence", "result", "finding" potentially show that scientific evidence is coming.

After loading the dataset and combining it with our sample dataset provided originally by the team leader we proceeded with the pre-processing actions on the data as already discussed before at the corresponding section. The first step of the baseline model was to classify the sentences of the abstracts into three categories (Claim, Evidence, Neither) by using the lexicons that we described above. This script is presented at the figure below.

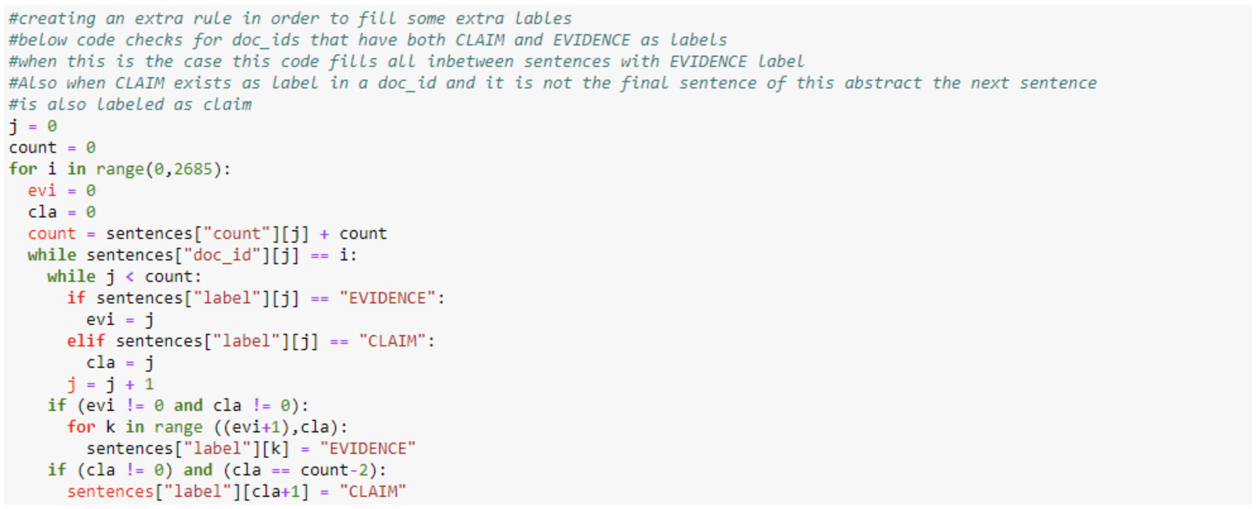


```
#labeling each sentence acoording to the lexicon rules that were created above
for i in range(0, len(sentences)):
  if sentences["sentence"].str.contains('|'.join(check_claim))[i]:
    sentences["label"][i] = "CLAIM"
  elif sentences["sentence"].str.contains('|'.join(check_evidence))[i]:
    sentences["label"][i] = "EVIDENCE"
  else:
    sentences["label"][i] = "NEITHER"
```

Figure 9: Step 1 of baseline model, argumentation using lexicons.

Furthermore, based on the empirical rules that evidence are usually followed by claims and that claim arguments are usually found at the end sentences of the abstract we classify again some sentences on the two rules. We check each abstract that contained sentences with both claim and evidence arguments after the classification using the lexicons.

When this is the case, our model replaces arguments on sentences that are between a claim and an evidence argument and have been previously characterized as "Neither" to "Evidence". Then, the model checks whether a Claim argument exists on an abstract on the penultimate sentence and not the final one. Finally, it classifies the next final sentence of this abstract as "Claim" too. This whole process is presented below.

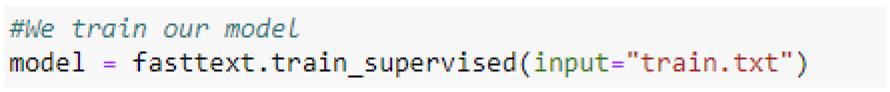


```
#creating an extra rule in order to fill some extra lables
#below code checks for doc_ids that have both CLAIM and EVIDENCE as labels
#when this is the case this code fills all inbetween sentences with EVIDENCE label
#Also when CLAIM exists as label in a doc_id and it is not the final sentence of this abstract the next sentence
#is also labeled as claim
j = 0
count = 0
for i in range(0,2685):
  evi = 0
  cla = 0
  count = sentences["count"][j] + count
  while sentences["doc_id"][j] == i:
    while j < count:
      if sentences["label"][j] == "EVIDENCE":
        evi = j
      elif sentences["label"][j] == "CLAIM":
        cla = j
      j = j + 1
    if (evi != 0 and cla != 0):
      for k in range ((evi+1),cla):
        sentences["label"][k] = "EVIDENCE"
    if (cla != 0) and (cla == count-2):
      sentences["label"][cla+1] = "CLAIM"
```

Figure 10: Step 2 of baseline model, argumentation using empirical rules.

Classification with Fasttext Approach

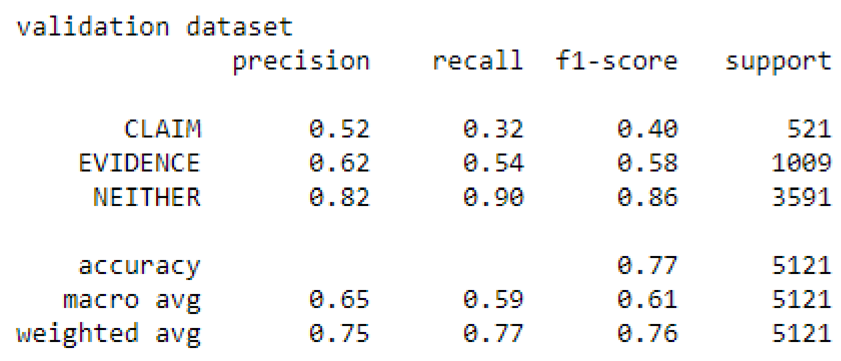
After following the already mentioned steps for text pre-processing we continued with labels classification using the Fasttext library. Before training our model, we split our dataset into three parts, train (60%), validation (20%) and test (20%) datasets. Furthermore, the text was processed and more specifically the sentence column in order to be transformed to the exact format that Fasttext algorithm require as input. Therefore, the "__label__" prefix was added before the argument. We then proceed with training our first model using as input the train dataset.



```
#We train our model
model = fasttext.train_supervised(input="train.txt")
```

Figure 11: Fasttext 1st model training.

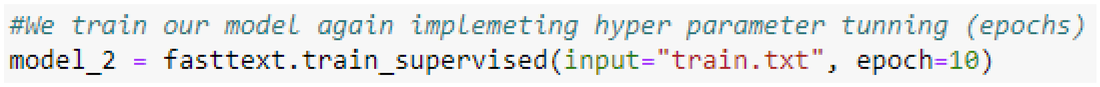
We proceeded with model evaluation on the validation dataset in order to get a first idea of some accuracy metrics like accuracy, precision and f1 score on our first model. The results of this are presented in the figure below.



```
validation dataset
              precision    recall  f1-score   support

       CLAIM       0.52      0.32      0.40       521
    EVIDENCE       0.62      0.54      0.58      1009
     NEITHER       0.82      0.90      0.86      3591

    accuracy                           0.77      5121
   macro avg       0.65      0.59      0.61      5121
weighted avg       0.75      0.77      0.76      5121
```

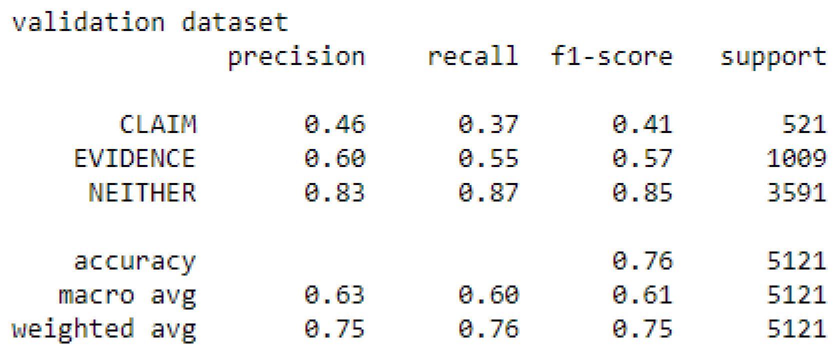
Figure 12: Evaluation of 1st model.

As already mentioned before, Fasttext algorithm supervised training has several hyper-parameters like Epochs, Learning Rate, etc., which need tunning in order for the model to produce better results. By default, Fasttext checks each training example only five times during training (epoch = 5), which is pretty small, given that our training set only have 20.5k training examples. The number of times each example is seen (also known as the number of epochs), can be increased using the -epoch option. In our 2nd model we tried to increase the epochs to 10 and evaluate again using our validation dataset.



```
#We train our model again implemeting hyper parameter tunning (epochs)
model_2 = fasttext.train_supervised(input="train.txt", epoch=10)
```

Figure 13: Fasttext 2nd model training (epochs).

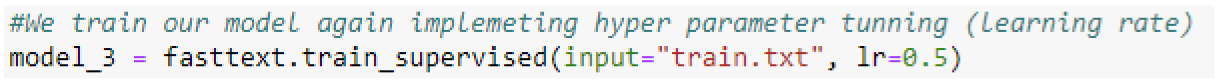


```
validation dataset
              precision    recall  f1-score   support

       CLAIM       0.46      0.37      0.41       521
    EVIDENCE       0.60      0.55      0.57      1009
     NEITHER       0.83      0.87      0.85      3591

    accuracy                           0.76      5121
   macro avg       0.63      0.60      0.61      5121
weighted avg       0.75      0.76      0.75      5121
```

Figure 14: Evaluation of 2nd model.

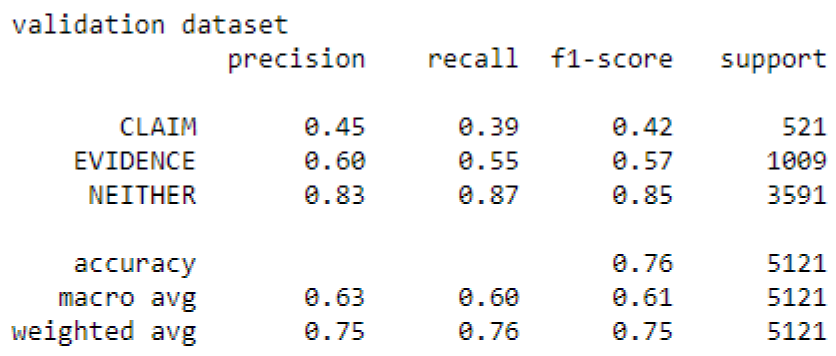
As we can observe precision went down as recall went up, resulting in a reduced f1 score. Accuracy of the model was also reduced slightly. By trying to increase epochs to a number greater than 10 the model produced worse results.

Another way to change the learning speed of our model is to increase (or decrease) the learning rate of the algorithm. This corresponds to how much the model changes after processing each example. Since the default learning rate of the algorithm is 0.1, we tried to increase it to 0.5 and train a 3rd model.



```
#We train our model again implemeting hyper parameter tunning (learning rate)
model_3 = fasttext.train_supervised(input="train.txt", lr=0.5)
```

Figure 15: Fasttext 3rd model training (learning rate)

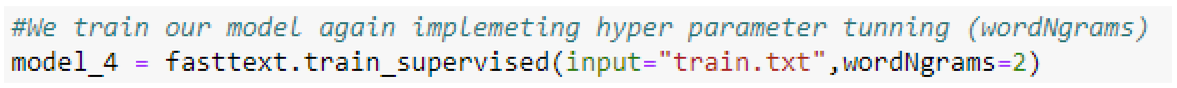


```
validation dataset
              precision    recall  f1-score   support

       CLAIM       0.45      0.39      0.42       521
    EVIDENCE       0.60      0.55      0.57      1009
     NEITHER       0.83      0.87      0.85      3591

    accuracy                           0.76      5121
   macro avg       0.63      0.60      0.61      5121
weighted avg       0.75      0.76      0.75      5121
```

Figure 16: Evaluation of 3rd model.

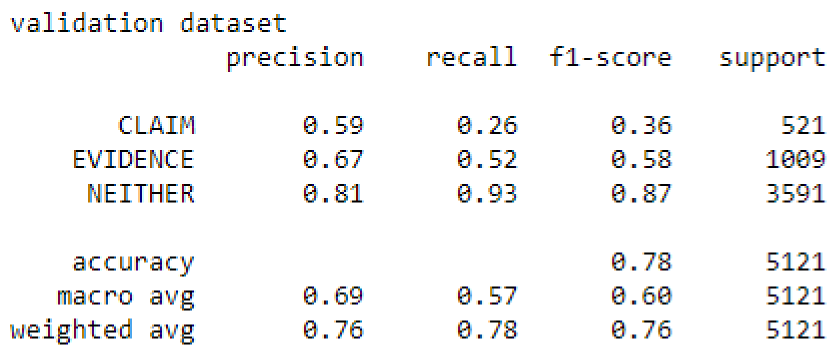
Learning's rate (lr) increment does not seem to make our model better, thus we will keep learning rate = 0.1 (by default)

Finally, we can improve the performance of a model by using word bigrams, instead of just unigrams. We train our 4th model.



```
#We train our model again implemeting hyper parameter tunning (wordNgrams)
model_4 = fasttext.train_supervised(input="train.txt",wordNgrams=2)
```

Figure 17: Fasttext 4th model training (word bigrams)

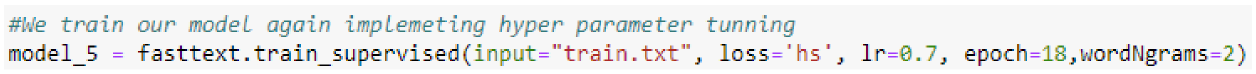


```
validation dataset
              precision    recall  f1-score   support

       CLAIM       0.59      0.26      0.36       521
    EVIDENCE       0.67      0.52      0.58      1009
     NEITHER       0.81      0.93      0.87      3591

    accuracy                           0.78      5121
   macro avg       0.69      0.57      0.60      5121
weighted avg       0.76      0.78      0.76      5121
```

Figure 18: Evaluation of 4th model.

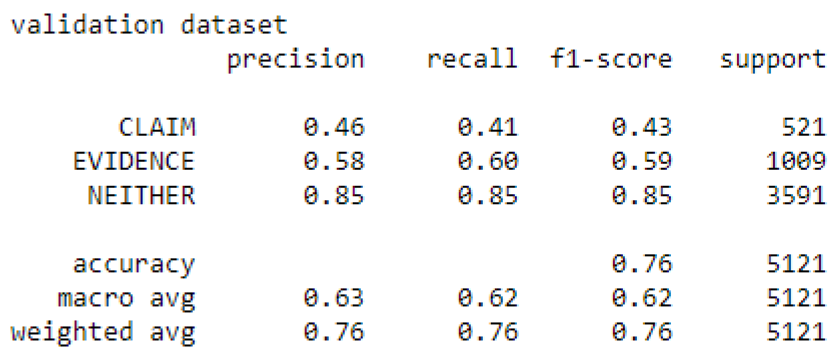
Word bigrams increment decreases f1 score as accuracy remains the same.

Finally, we will try to combine all above hyper-parameters to a 5th model and evaluate it. We, therefore, tried different values for these hyperparameter and concluded to set epochs = 18, learning rate = 0.7, word bigrams = 2 and the loss function to hierarchical SoftMax. This model and its evaluation are presented in the figures below.



```
#We train our model again implemeting hyper parameter tunning
model_5 = fasttext.train_supervised(input="train.txt", loss='hs', lr=0.7, epoch=18,wordNgrams=2)
```

Figure 19: Fasttext final model training(combination of hyper-parameter tunning)

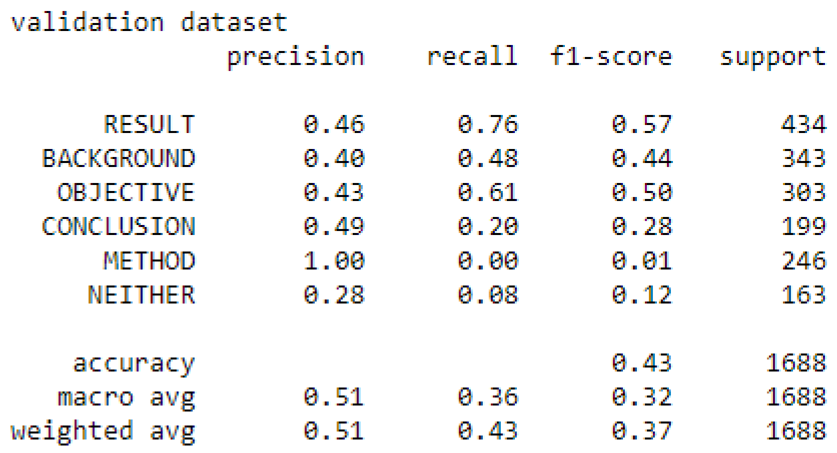


```
validation dataset
              precision    recall  f1-score   support

       CLAIM       0.46      0.41      0.43       521
    EVIDENCE       0.58      0.60      0.59      1009
     NEITHER       0.85      0.85      0.85      3591

    accuracy                           0.76      5121
   macro avg       0.63      0.62      0.62      5121
weighted avg       0.76      0.76      0.76      5121
```

Figure 20: Evaluation of final model.

As we can observe this model produces slightly better f1 scores as the accuracy remains nearly the same. Thus, we decided to proceed with this model and test it on our test dataset which will be presented later.

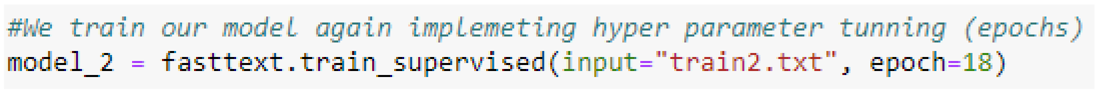
The same procedure was followed using the Fasttext algorithm with the dataset that contained "structure labels" as input. We will start with presenting the results for the basic model without hyper-parameter tunning.



```
validation dataset
              precision    recall  f1-score   support

      RESULT       0.46      0.76      0.57       434
  BACKGROUND       0.40      0.48      0.44       343
   OBJECTIVE       0.43      0.61      0.50       303
  CONCLUSION       0.49      0.20      0.28       199
      METHOD       1.00      0.00      0.01       246
     NEITHER       0.28      0.08      0.12       163

    accuracy                           0.43      1688
   macro avg       0.51      0.36      0.32      1688
weighted avg       0.51      0.43      0.37      1688
```

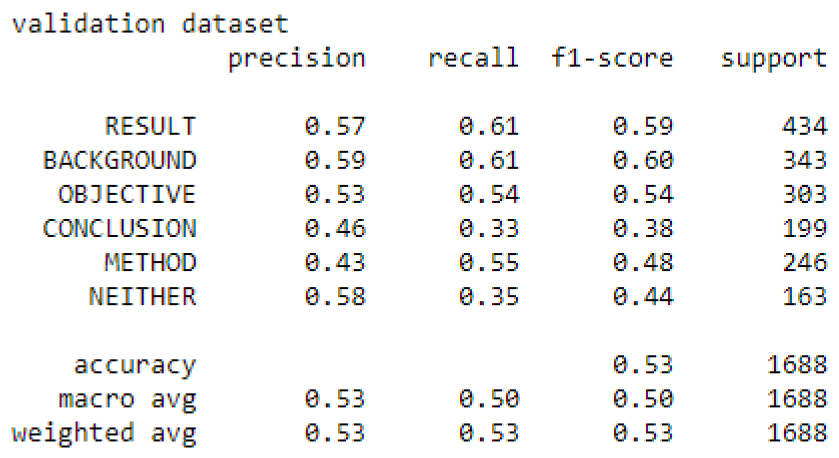
Figure 21: Evaluation of basic model on dataset with structure labels.

We then tried to increase the epochs to 18 and trained our 2nd model.



```
#We train our model again implemeting hyper parameter tunning (epochs)
model_2 = fasttext.train_supervised(input="train2.txt", epoch=18)
```

Figure 22: Training of the 2nd model on dataset with structure labels

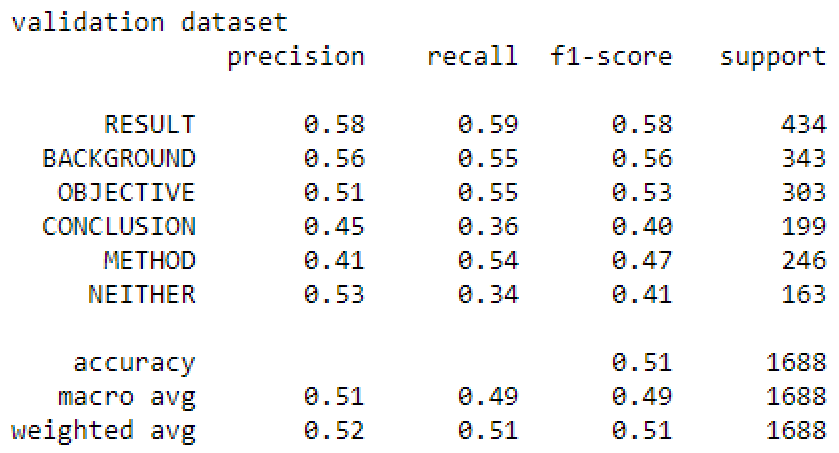


```
validation dataset
              precision    recall  f1-score   support

      RESULT       0.57      0.61      0.59       434
  BACKGROUND       0.59      0.61      0.60       343
   OBJECTIVE       0.53      0.54      0.54       303
  CONCLUSION       0.46      0.33      0.38       199
      METHOD       0.43      0.55      0.48       246
     NEITHER       0.58      0.35      0.44       163

    accuracy                           0.53      1688
   macro avg       0.53      0.50      0.50      1688
weighted avg       0.53      0.53      0.53      1688
```

Figure 23: Evaluation of 2nd model on dataset with structure labels.

As we can observe epoch's increment does make our model better and more specifically by increasing f1 scores and accuracy from 0.43 to 0.53.

We then tried to make our model better by changing the learning rate (lr = 0.5). The results of our 3rd model are presented below.

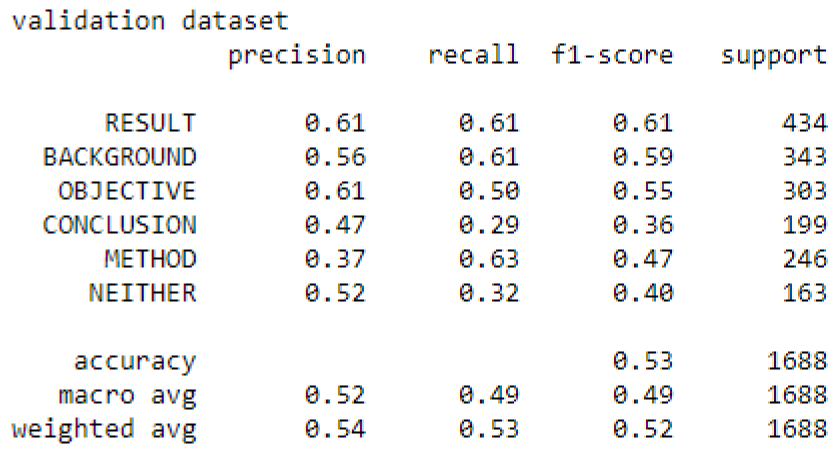


```
validation dataset
              precision    recall  f1-score   support

      RESULT       0.58      0.59      0.58       434
  BACKGROUND       0.56      0.55      0.56       343
   OBJECTIVE       0.51      0.55      0.53       303
  CONCLUSION       0.45      0.36      0.40       199
      METHOD       0.41      0.54      0.47       246
     NEITHER       0.53      0.34      0.41       163

    accuracy                           0.51      1688
   macro avg       0.51      0.49      0.49      1688
weighted avg       0.52      0.51      0.51      1688
```

Figure 24: Evaluation of 3rd model on dataset with structure labels.

Learning's rate (lr) increment does not seem to make our model better as accuracy and f1 scores are reduces, thus we will set the learning rate = 0.1 which is the default value.

We continued by using word bigrams instead of unigrams as we did before. The results of the 4th model are presented below.

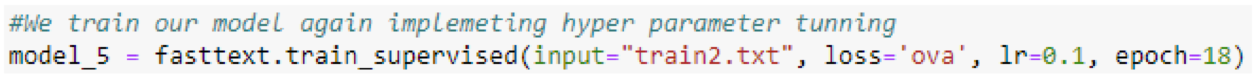


```
validation dataset
              precision    recall  f1-score   support

      RESULT       0.61      0.61      0.61       434
  BACKGROUND       0.56      0.61      0.59       343
   OBJECTIVE       0.61      0.50      0.55       303
  CONCLUSION       0.47      0.29      0.36       199
      METHOD       0.37      0.63      0.47       246
     NEITHER       0.52      0.32      0.40       163

    accuracy                           0.53      1688
   macro avg       0.52      0.49      0.49      1688
weighted avg       0.54      0.53      0.52      1688
```

Figure 25: Evaluation of 4th model on dataset with structure labels.

WordNgrams increment does not make our model better, as it produces similar results on f1 score and accuracy.

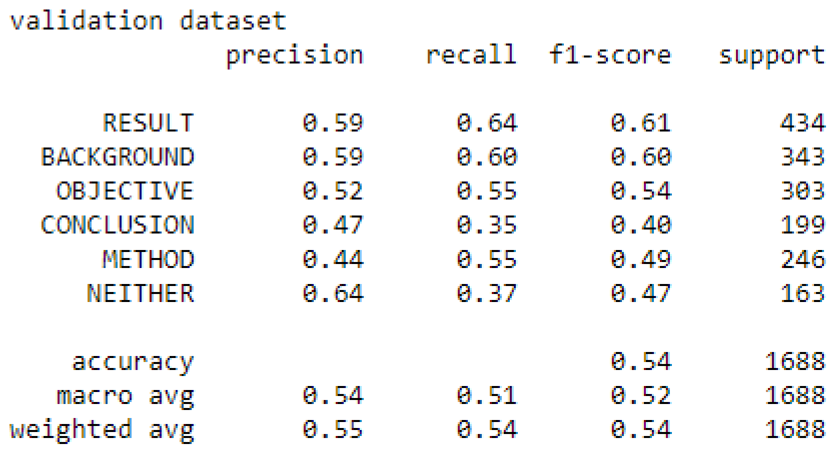
Finally, we form our final model by choosing the best combination of these hyper-parameters. The final model is:



```
#We train our model again implemeting hyper parameter tunning
model_5 = fasttext.train_supervised(input="train2.txt", loss='ova', lr=0.1, epoch=18)
```

Figure 26: Training of the final model on dataset with structure labels

This time we included another loss function to our model and more specifically one which used independent binary classifiers for each label, since this one seem to produce the best results. Final model's results are shown at below figure.



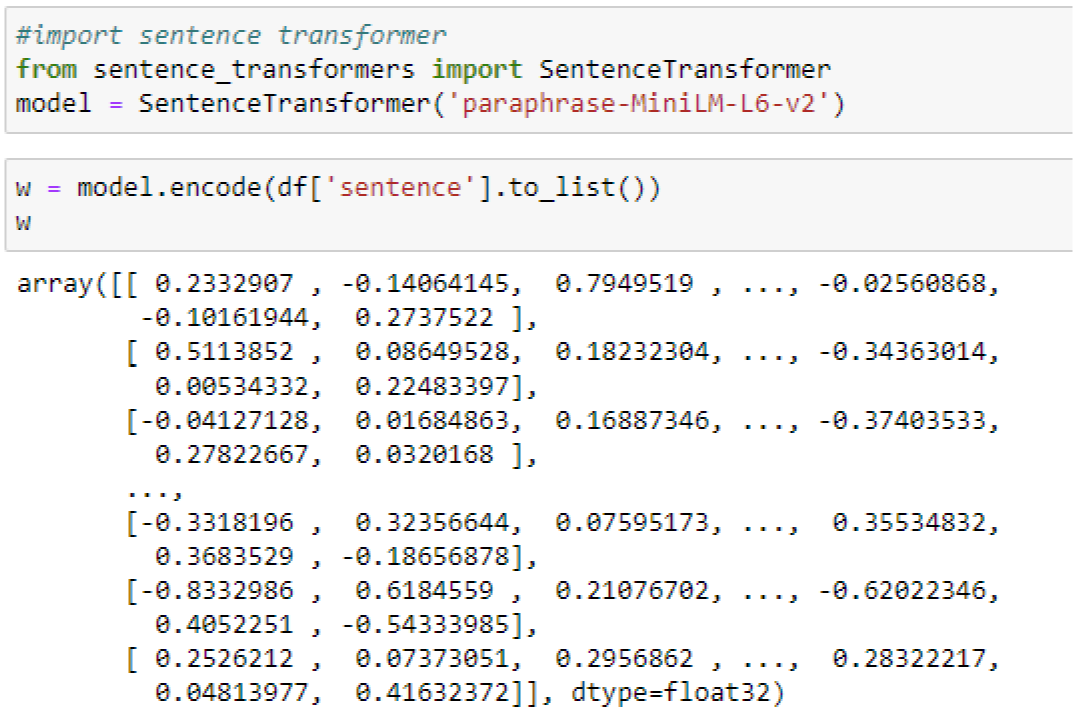
```
validation dataset
              precision    recall  f1-score   support

      RESULT       0.59      0.64      0.61       434
  BACKGROUND       0.59      0.60      0.60       343
   OBJECTIVE       0.52      0.55      0.54       303
  CONCLUSION       0.47      0.35      0.40       199
      METHOD       0.44      0.55      0.49       246
     NEITHER       0.64      0.37      0.47       163

    accuracy                           0.54      1688
   macro avg       0.54      0.51      0.52      1688
weighted avg       0.55      0.54      0.54      1688
```

Figure 27: Evaluation of final model on dataset with structure labels.

This model produced the best results on accuracy and f1 score thus we decided to proceed with this model and evaluate it on our test dataset.

Clustering using Graph

The first step of the clustering process after loading the datasets and finishing text pre-processing was to extract the sentence embeddings from the models. It should be noted that for the following process only sentences which contained arguments (Claim, Evidence) were used from the original dataset. Model was produced using the sentence-transformer library [7] and sentence embeddings were extracted and passed to a list with the command model.encode.



```
#import sentence transformer
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')
```

```
w = model.encode(df['sentence'].to_list())
w
```

```
array([[ 0.2332907 , -0.14064145,  0.7949519 , ..., -0.02560868,
        -0.10161944,  0.2737522 ],
       [ 0.5113852 ,  0.08649528,  0.18232304, ..., -0.34363014,
         0.00534332,  0.22483397],
       [-0.04127128,  0.01684863,  0.16887346, ..., -0.37403533,
         0.27822667,  0.0320168 ],
       ...,
       [-0.3318196 ,  0.32356644,  0.07595173, ...,  0.35534832,
         0.3683529 , -0.18656878],
       [-0.8332986 ,  0.6184559 ,  0.21076702, ..., -0.62022346,
         0.4052251 , -0.54333985],
       [ 0.2526212 ,  0.07373051,  0.2956862 , ...,  0.28322217,
         0.04813977,  0.41632372]], dtype=float32)
```

Figure 28: Sentence Embeddings

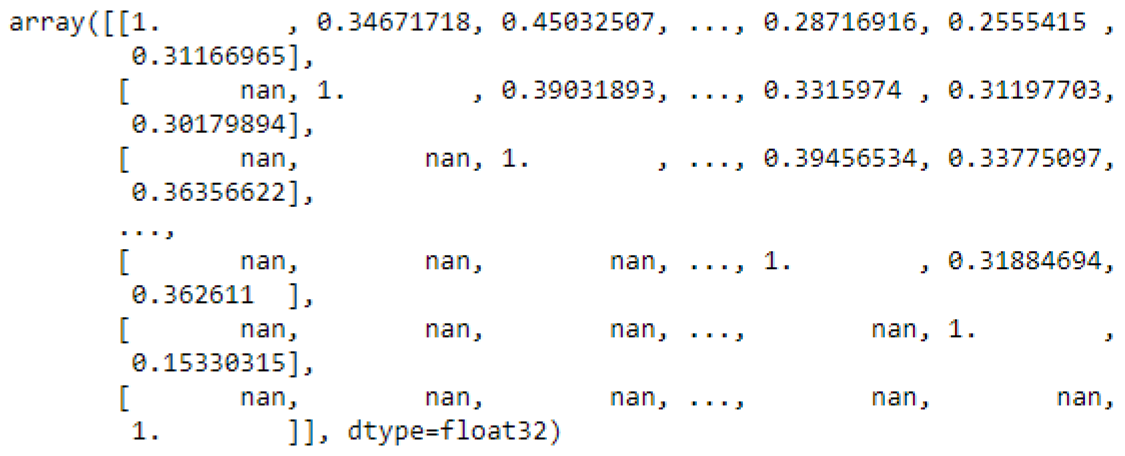
Sentence embeddings were then rescaled by following the min-max normalization which formula is:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

We proceed with creating the distance matrix in order to get have a metric that connects each argument. In order to do this the sklearn library was used and more specifically the function pairwise_distances from sklearn.metrics. This method takes either a vector array or a distance matrix and returns a distance matrix. After creating the distance matrix, we normalized it using the same method that was mentioned above. From the distance matrix we decided to proceed with creating a similarity matrix using the formula:

$$similarity = 1 - distance$$

The final similarity matrix after keeping only upper triangular values looked like this:



```
array([[1.        , 0.34671718, 0.45032507, ..., 0.28716916, 0.2555415 ,
        0.31166965],
       [       nan, 1.        , 0.39031893, ..., 0.3315974 , 0.31197703,
        0.30179894],
       [       nan,        nan, 1.        , ..., 0.39456534, 0.33775097,
        0.36356622],
       ...,
       [       nan,        nan,        nan, ..., 1.        , 0.31884694,
        0.362611  ],
       [       nan,        nan,        nan, ...,        nan, 1.        ,
        0.15330315],
       [       nan,        nan,        nan, ...,        nan,        nan,
        1.        ]], dtype=float32)
```

Figure 29: Similarity Matrix

A basic exploratory data analysis was conducted on the similarity matrix. Below we can see the histogram of the similarity matrix.

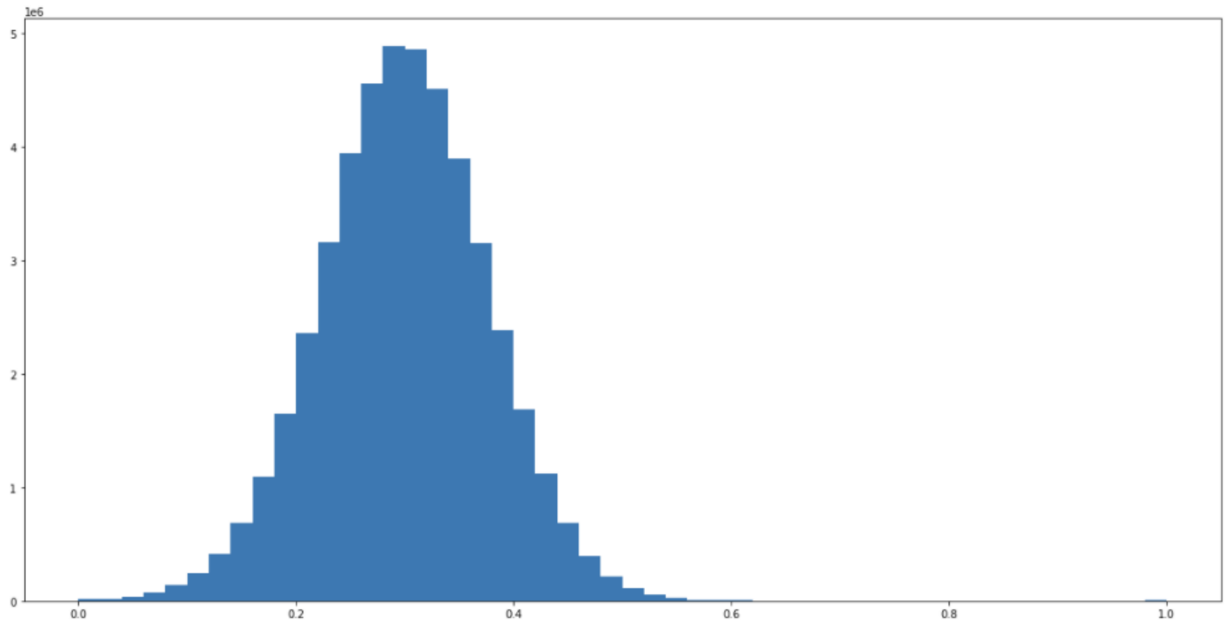


Figure 30: Histogram of similarity matrix

Since this histogram seems to follow the normal distribution, we calculated the mean and standard deviation.

$$mean = 0.3$$

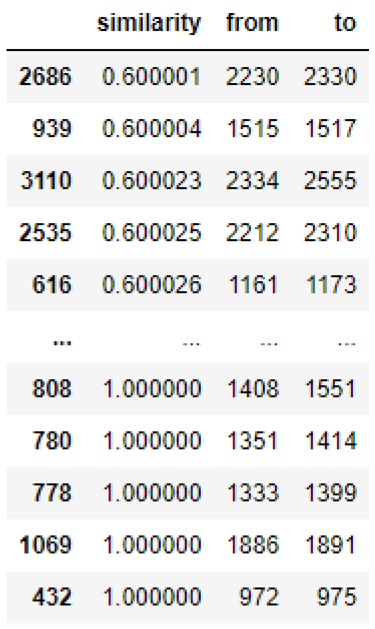
$$standard\ deviation = 0.08$$

Normally, since this is following the normal distribution the values between mean ± 3σ must be kept. After trying to keep only the values with similarity > 0.38 the graph that was produced was too dense with a lot of edges connecting different articles, therefore creating giant clique and community components. We therefore decided to pick the threshold according to the observations number of each bin in the histogram which are presented in below matrix.

Number of observations	Similarity range
23315511	0 – 0.1
2044619	0.1 – 0.2
9443238	0.2 – 0.3
9397561	0.3 – 0.4
2049584	0.4 – 0.5
100369	0.5 – 0.6
2912	0.6 – 0.7
163	0.7 – 0.8
18	0.8 – 0.9
4843	0.9 – 1

At this point we proceeded with similarities > 0.6 in order to produce a better classification. It must also be noted that 4843 which is the number of observations having range between 0.9 – 1 is such a huge number because of all the links between the arguments of the same articles which will be removed at a next step.

From this similarity matrix with values only > 0.6 we produced a list which contained the index of the similarity matrix and the similarity between them. Later these indexes were transformed to the corresponding articles ids for each sentence. Finally, we removed links with similarities between sentences of the same id. In the case where multiple links between same article ids were present, we kept the one with the highest similarity value. The final coordinates of links that will be later be used as edges are presented as a data frame below.



	similarity	from	to
2686	0.600001	2230	2330
939	0.600004	1515	1517
3110	0.600023	2334	2555
2535	0.600025	2212	2310
616	0.600026	1161	1173
...	...	...	...
808	1.000000	1408	1551
780	1.000000	1351	1414
778	1.000000	1333	1399
1069	1.000000	1886	1891
432	1.000000	972	975

Figure 31: Final links between articles based on similarities

As a final step the graph was created using the NetworkX library in python [8]. All articles ids were added as nodes and we created edges between them using the links that were created and presented above as input. The whole graph is presented below.

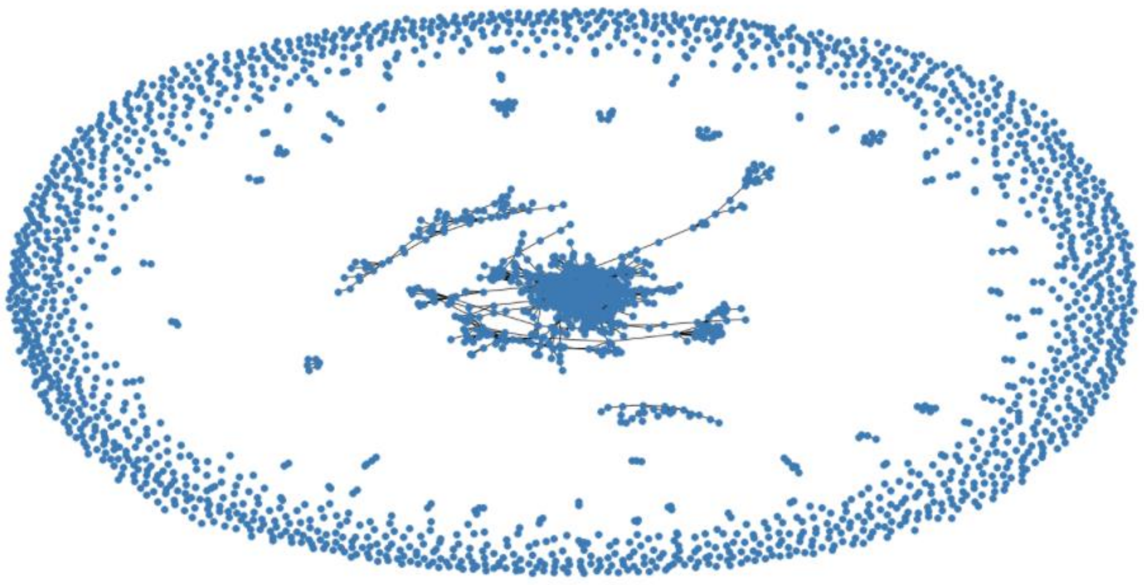


Figure 32: Graph plot

We started visualizing the communities of k-cliques where k is the number of least nodes that are all connected with each other.

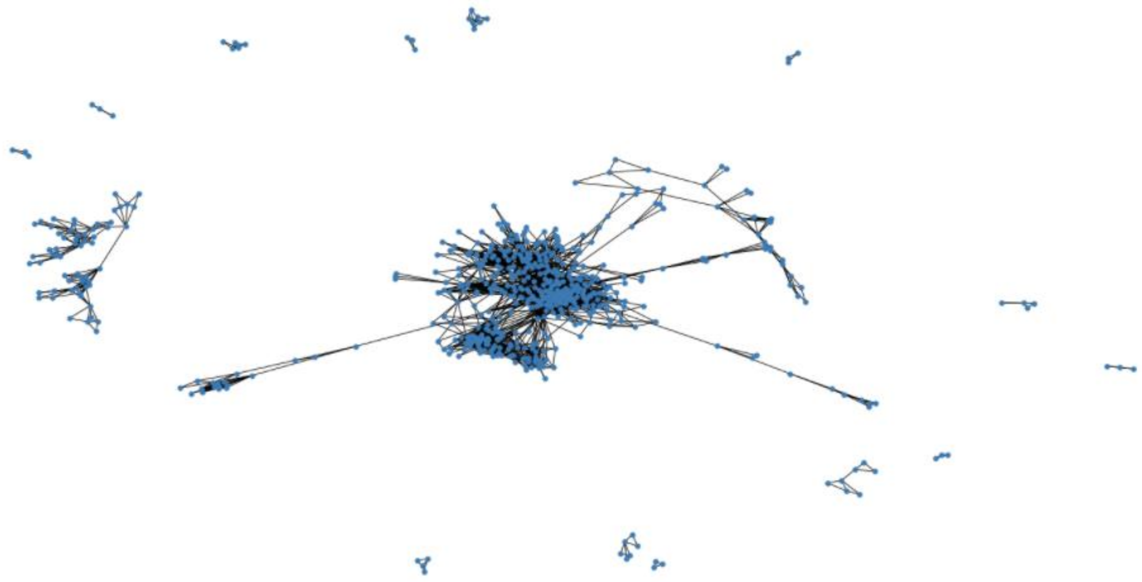


Figure 33: Communities with k=3 (similarity >0.6)

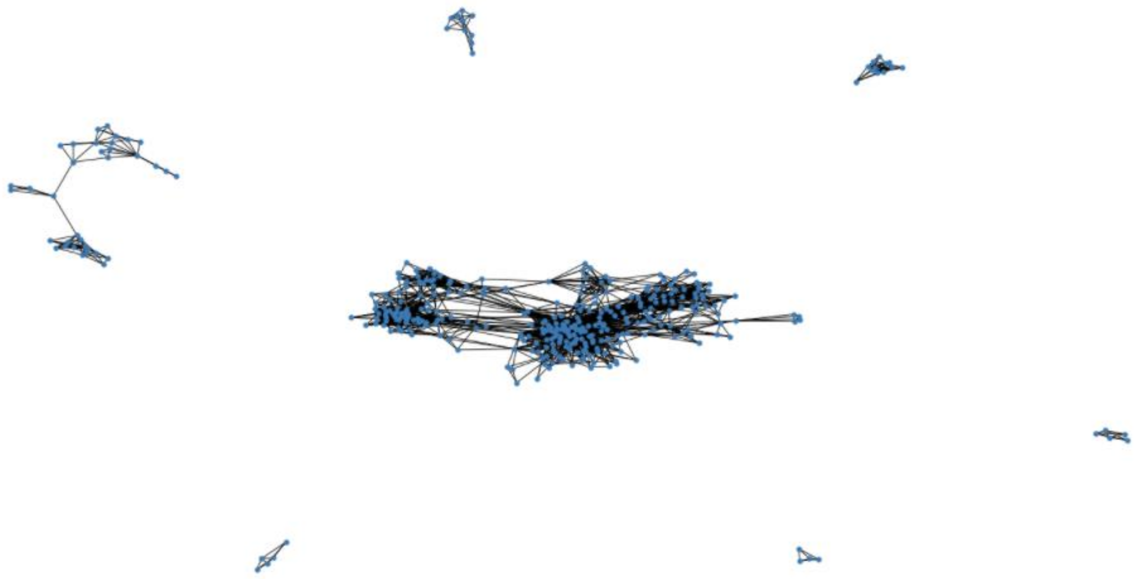


Figure 34: Communities with k=4 (similarity >0.6)

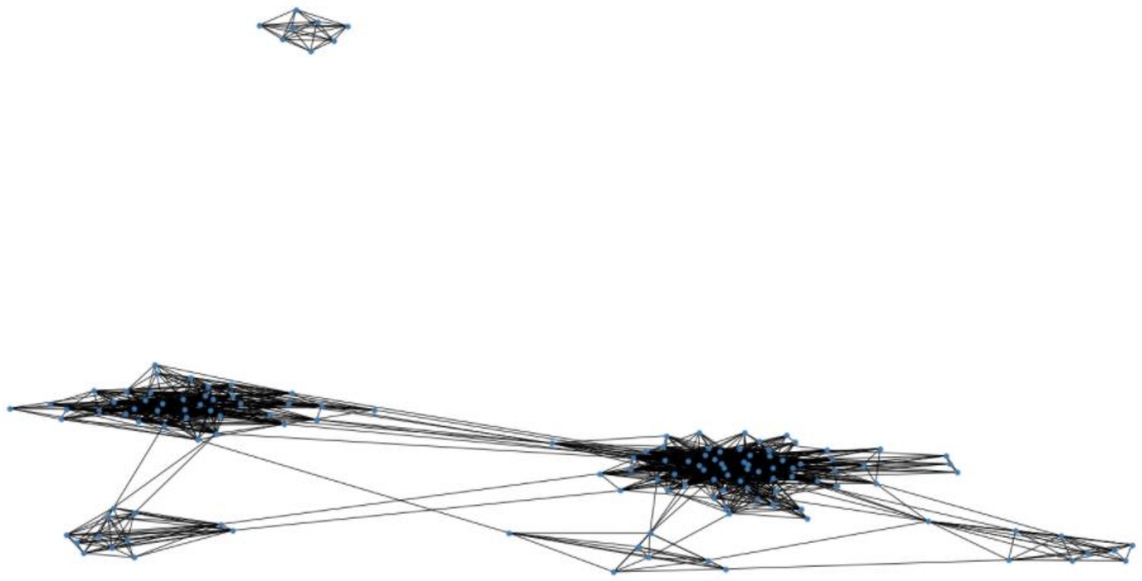


Figure 35: Communities with k=7 (similarity >0.6)

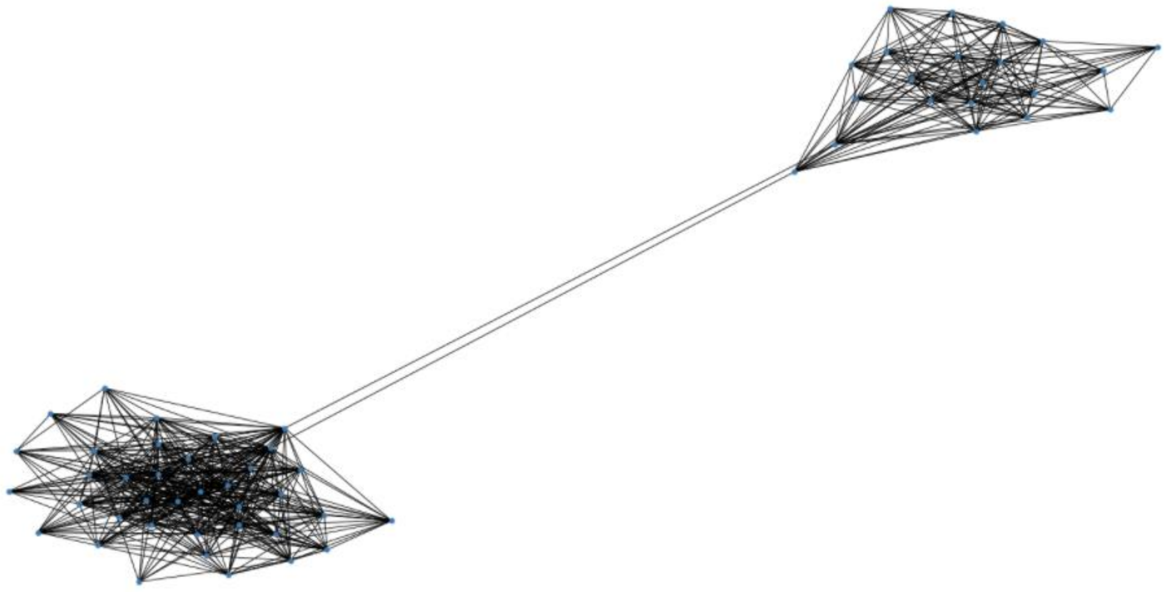


Figure 36: Communities with k=10 (similarity >0.6)

As we can observe, bigger communities appear as k is increased. Since we needed a classification where the links (edges) between articles will have higher similarity, we dropped all edges with similarity < 0.65 and visualize the communities again.

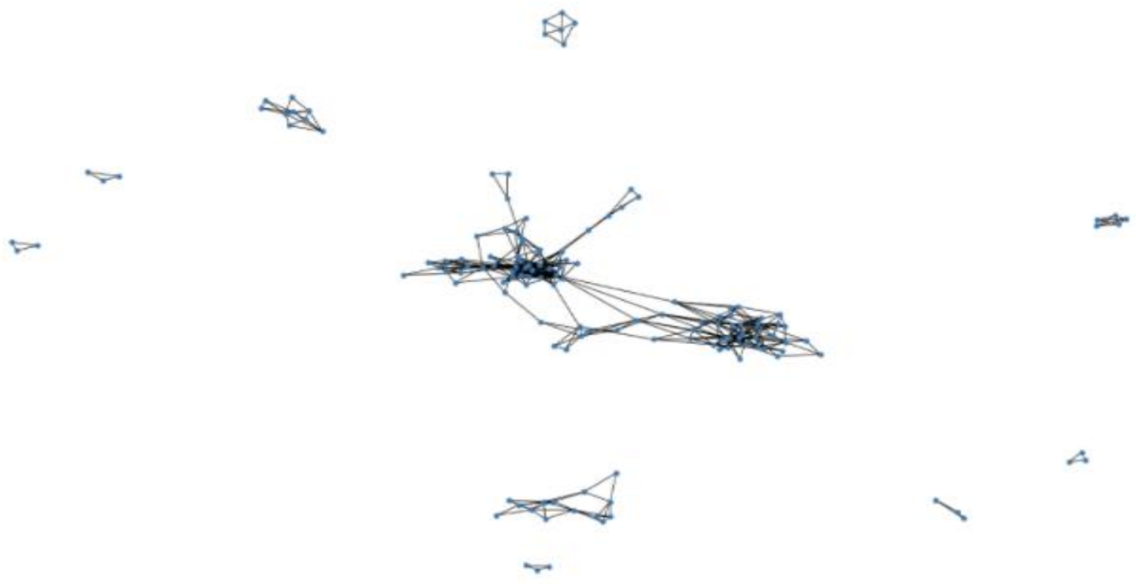


Figure 37: Communities with k=3 (similarity >0.65)

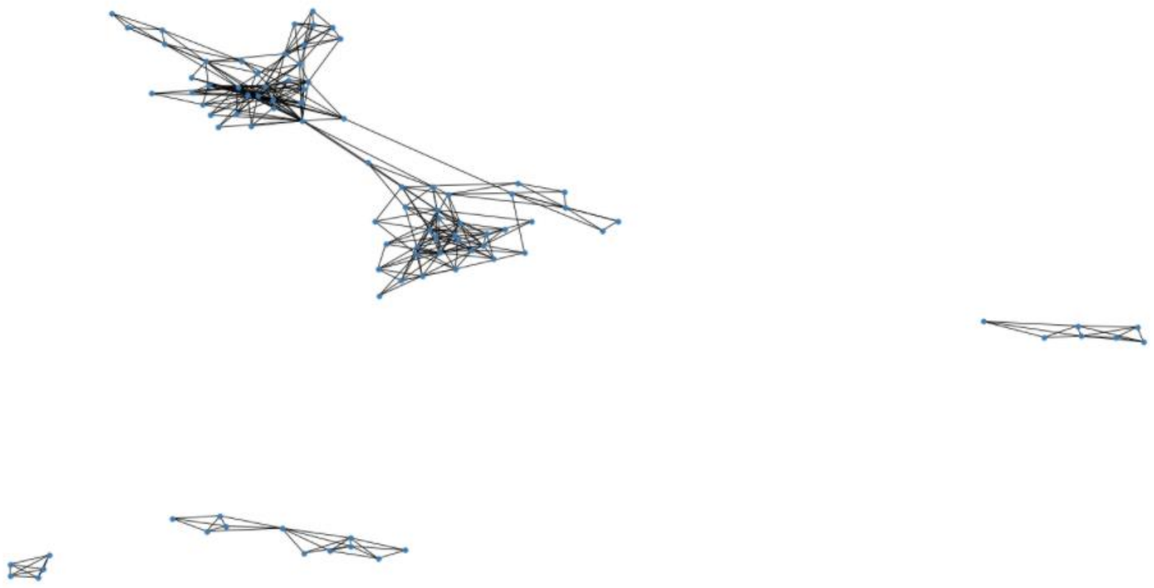


Figure 38: Communities with k=4 (similarity >0.65)

We observed that the communities that are formed started getting less dense. We experiment with dropping some more edges and keep only edges with similarity > 0.7 and plot our final visualizations.

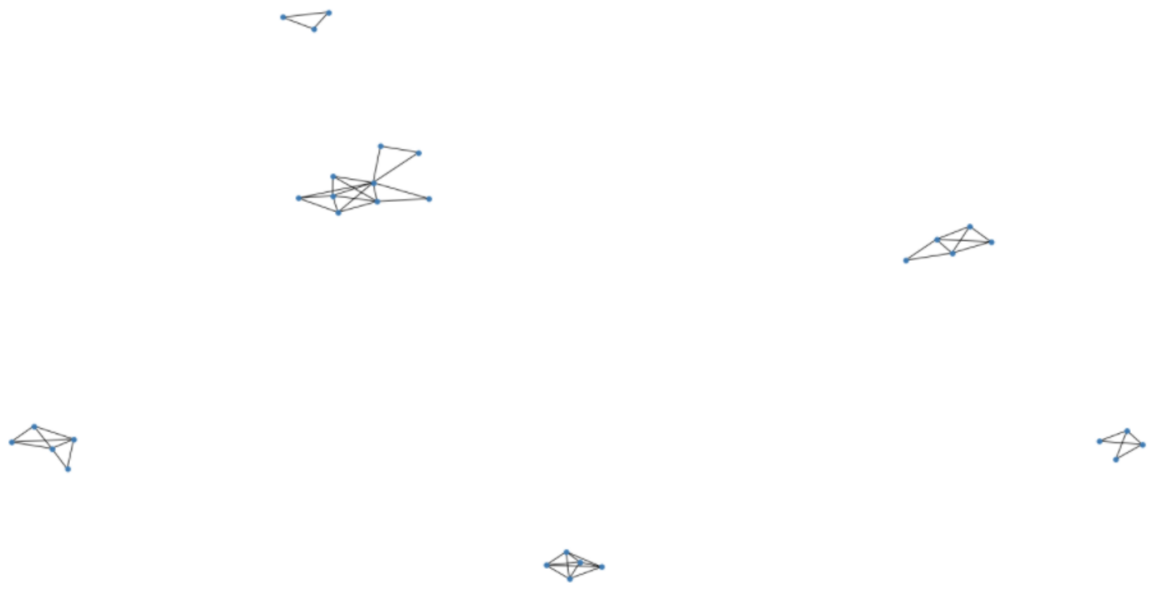


Figure 39: Communities with k=3 (similarity >0.7)

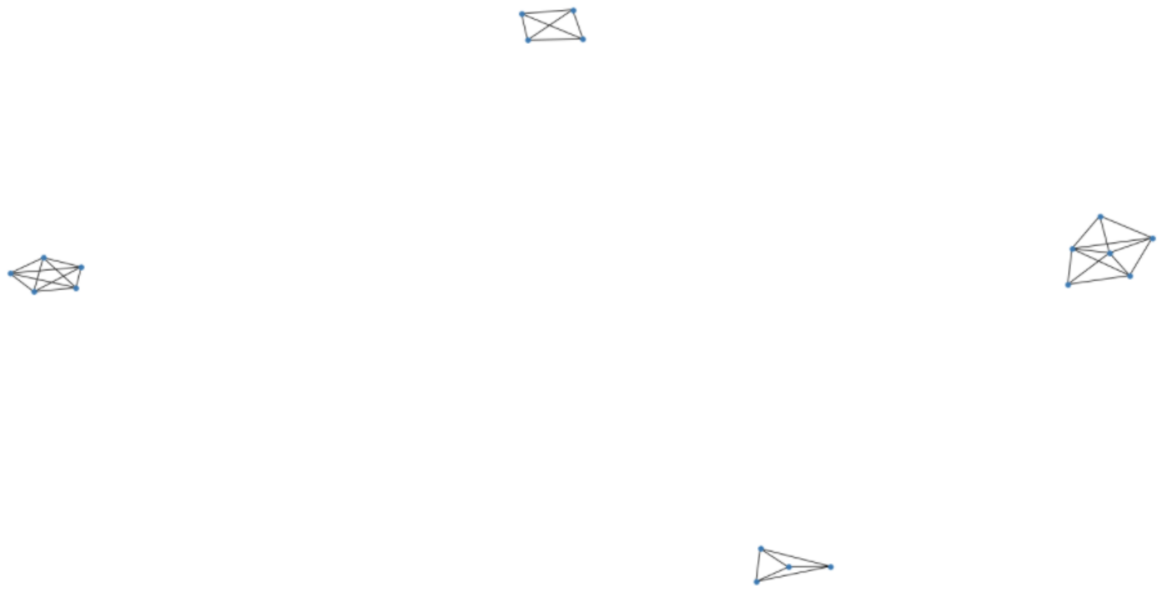


Figure 40: Communities with k=4 (similarity >0.7)

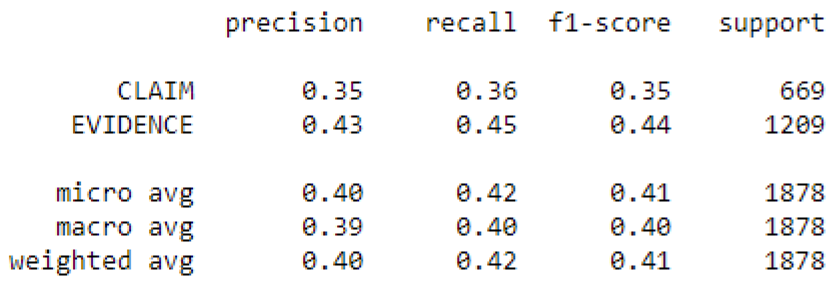
The communities continue to get less dense as k increases. We will finally check our classification and clustering models in the next section where the results will be presented.

It must also be noted that another attempt at clustering the abstracts with the graph method. In this case only sentences that were labeled with a “Claim” argument was taken as input when creating the sentence embeddings. The communities formed with the same thresholds and parameter ok k were plotted and are presented in the appendix. In conclusion this clustering produced less number of communities so it was decided that the best clustering model was using “Claim” and “Evidence” arguments as sentence embeddings.

3.2 Results & Quantitative Analysis (incl. visualizations)

Intuitive Baseline

According to the Intuitive Baseline model’s results we can observe some metrics as precision, recall and f1-score.



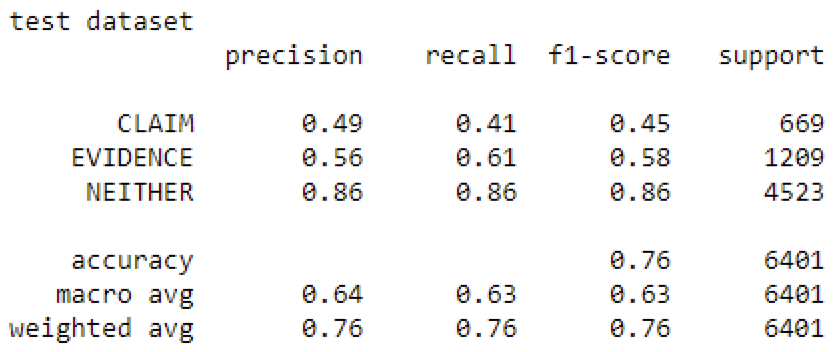
```
              precision    recall  f1-score   support

       CLAIM       0.35      0.36      0.35       669
    EVIDENCE       0.43      0.45      0.44      1209

   micro avg       0.40      0.42      0.41      1878
   macro avg       0.39      0.40      0.40      1878
weighted avg       0.40      0.42      0.41      1878
```

Figure 41: Intuitive Baseline results

Classification using Fasttext Approach

- Argumentative classification

The corresponding metrics can also be observed for the classification using Fasttext approach and we can observe that our model has better performance for the same test dataset. As figure [] depicts, f1-score has been well improved in comparison with the corresponding f1 score of Intuitive Baseline’s approach.

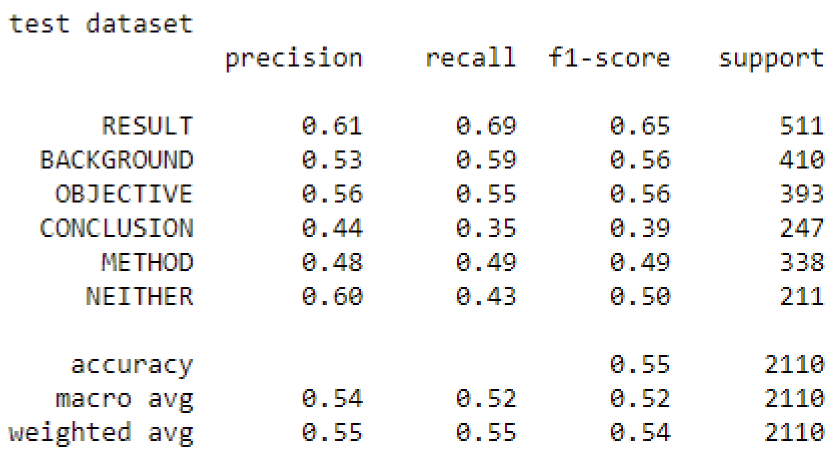


```
test dataset
              precision    recall  f1-score   support

       CLAIM       0.49      0.41      0.45       669
    EVIDENCE       0.56      0.61      0.58      1209
     NEITHER       0.86      0.86      0.86      4523

    accuracy                           0.76      6401
   macro avg       0.64      0.63      0.63      6401
weighted avg       0.76      0.76      0.76      6401
```

Figure 42: Argumentative classification results - Fasttext Approach

- Structure classification

As we can observe the structure classification model does not perform as well as the argumentative one as its accuracy is lower. This effect might be related with the fact that in the argumentative classification case, the number of sentences labeled as ‘NEITHER’ form by far the majority of our dataset.



```
test dataset
              precision    recall  f1-score   support

      RESULT       0.61      0.69      0.65       511
  BACKGROUND       0.53      0.59      0.56       410
   OBJECTIVE       0.56      0.55      0.56       393
  CONCLUSION       0.44      0.35      0.39       247
      METHOD       0.48      0.49      0.49       338
     NEITHER       0.60      0.43      0.50       211

    accuracy                           0.55      2110
   macro avg       0.54      0.52      0.52      2110
weighted avg       0.55      0.55      0.54      2110
```

Figure 43: Structure classification results - Fasttext Approach

- ***Abstracts' clustering using Graph method***

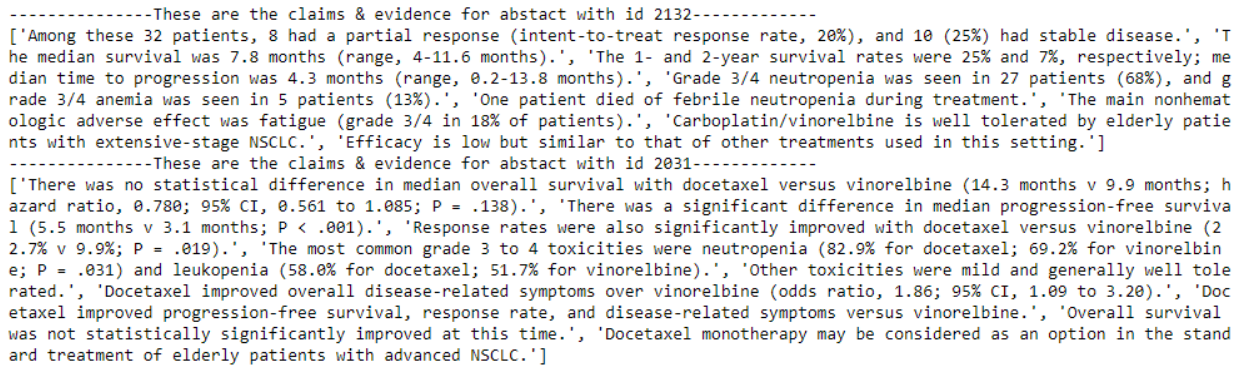
As already described in section 3.1 the Graph method achieved the desired goal. We managed to cluster similar abstracts together and form communities which we visualized for different k-cliques in section 3.1. Some examples of visualizations which show a different number of communities depending on k-value can be observed in [Figure 33, Figure 34, Figure 35, Figure 36, Figure 37, Figure 38, Figure 39, Figure 40].

3.3 Qualitative & Error Analysis

As far as it concerns the Intuitive Baseline classification model, we can argue that the model can be characterized as overfitted. As we previously described, the model tense to classify all sentences between evidence and a claim argument in the same abstract as evidence too. Also, when the model detects a claim argument which is followed by another sentence, then, this sentence is also characterized as claim argument. Therefore, the model classifies many more sentences as arguments than it should.

Furthermore, regarding the Fasttext classification for the argumentative statements, the distribution of argument labels is skewed as the sample is imbalanced. Particularly, ‘NEITHER’ labels constitute the large majority of the dataset, so the classifiers tend to easily characterize an argument as ‘NEITHER’. Consequently, its results are characterized as biased.

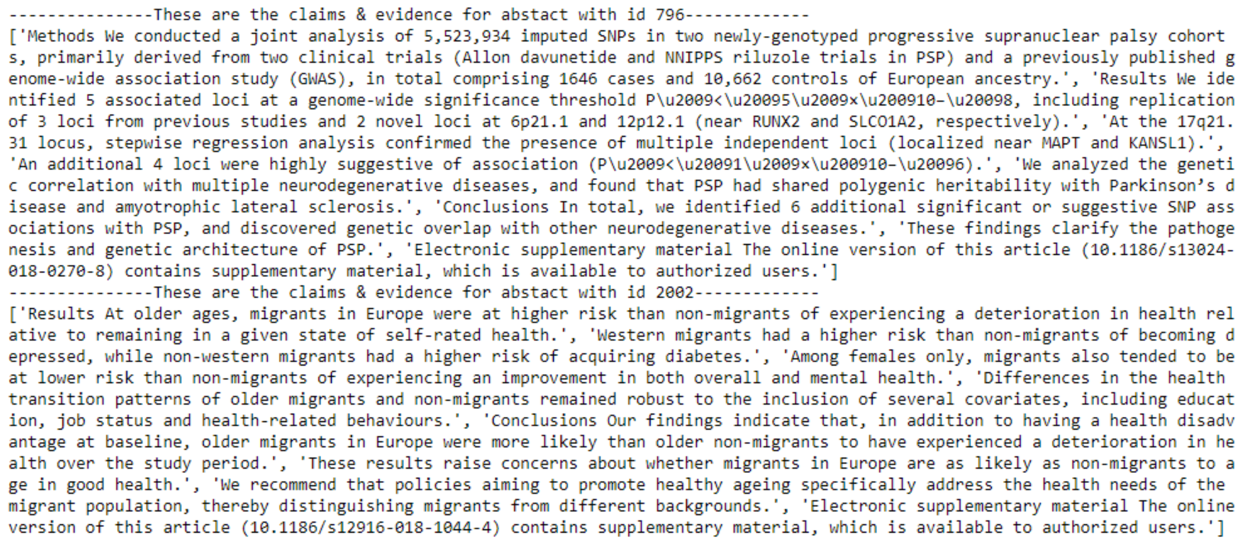
Finally, the annotation process can be strongly related with the errors occurred in the clustering procedure using the Graph method. As pictures below show, there are many abstracts wrongly annotated. As we can see in figure 44, these two abstracts are correctly clustered as they both refer to treatments for elderly patients. In contrary, figure 45 depicts abstracts that are wrongly clustered together only because the last sentence was annotated as claim wrongly.



```
---------------These are the claims & evidence for abstact with id 2132-------------
['Among these 32 patients, 8 had a partial response (intent-to-treat response rate, 20%), and 10 (25%) had stable disease.', 'T
he median survival was 7.8 months (range, 4-11.6 months).', 'The 1- and 2-year survival rates were 25% and 7%, respectively; me
dian time to progression was 4.3 months (range, 0.2-13.8 months).', 'Grade 3/4 neutropenia was seen in 27 patients (68%), and g
rade 3/4 anemia was seen in 5 patients (13%).', 'One patient died of febrile neutropenia during treatment.', 'The main nonhemat
ologic adverse effect was fatigue (grade 3/4 in 18% of patients).', 'Carboplatin/vinorelbine is well tolerated by elderly patie
nts with extensive-stage NSCLC.', 'Efficacy is low but similar to that of other treatments used in this setting.']
---------------These are the claims & evidence for abstact with id 2031-------------
['There was no statistical difference in median overall survival with docetaxel versus vinorelbine (14.3 months v 9.9 months; h
azard ratio, 0.780; 95% CI, 0.561 to 1.085; P = .138).', 'There was a significant difference in median progression-free surviva
l (5.5 months v 3.1 months; P < .001).', 'Response rates were also significantly improved with docetaxel versus vinorelbine (2
2.7% v 9.9%; P = .019).', 'The most common grade 3 to 4 toxicities were neutropenia (82.9% for docetaxel; 69.2% for vinorelbin
e; P = .031) and leukopenia (58.0% for docetaxel; 51.7% for vinorelbine).', 'Other toxicities were mild and generally well tole
rated.', 'Docetaxel improved overall disease-related symptoms over vinorelbine (odds ratio, 1.86; 95% CI, 1.09 to 3.20).', 'Doc
etaxel improved progression-free survival, response rate, and disease-related symptoms versus vinorelbine.', 'Overall survival
was not statistically significantly improved at this time.', 'Docetaxel monotherapy may be considered as an option in the stand
ard treatment of elderly patients with advanced NSCLC.']
```

Figure 44: Qualitive error analysis 1.

These two sentences have the exact same format and words so, their embeddings were too similar. Thus, better annotation would probably lead to better clustering model.



```
---------------These are the claims & evidence for abstact with id 796-------------
['Methods We conducted a joint analysis of 5,523,934 imputed SNPs in two newly-genotyped progressive supranuclear palsy cohort
s, primarily derived from two clinical trials (Allon davunetide and NNIPPS riluzole trials in PSP) and a previously published g
enome-wide association study (GWAS), in total comprising 1646 cases and 10,662 controls of European ancestry.', 'Results We ide
ntified 5 associated loci at a genome-wide significance threshold P\u2009<\u20095\u2009×\u200910–\u20098, including replication
of 3 loci from previous studies and 2 novel loci at 6p21.1 and 12p12.1 (near RUNX2 and SLCO1A2, respectively).', 'At the 17q21.
31 locus, stepwise regression analysis confirmed the presence of multiple independent loci (localized near MAPT and KANSL1).',
'An additional 4 loci were highly suggestive of association (P\u2009<\u20091\u2009×\u200910–\u20096).', 'We analyzed the geneti
c correlation with multiple neurodegenerative diseases, and found that PSP had shared polygenic heritability with Parkinson’s d
isease and amyotrophic lateral sclerosis.', 'Conclusions In total, we identified 6 additional significant or suggestive SNP ass
ociations with PSP, and discovered genetic overlap with other neurodegenerative diseases.', 'These findings clarify the pathoge
nesis and genetic architecture of PSP.', 'Electronic supplementary material The online version of this article (10.1186/s13024-
018-0270-8) contains supplementary material, which is available to authorized users.']
---------------These are the claims & evidence for abstact with id 2002-------------
['Results At older ages, migrants in Europe were at higher risk than non-migrants of experiencing a deterioration in health rel
ative to remaining in a given state of self-rated health.', 'Western migrants had a higher risk than non-migrants of becoming d
epressed, while non-western migrants had a higher risk of acquiring diabetes.', 'Among females only, migrants also tended to be
at lower risk than non-migrants of experiencing an improvement in both overall and mental health.', 'Differences in the health
transition patterns of older migrants and non-migrants remained robust to the inclusion of several covariates, including educat
ion, job status and health-related behaviours.', 'Conclusions Our findings indicate that, in addition to having a health disadv
antage at baseline, older migrants in Europe were more likely than older non-migrants to have experienced a deterioration in he
alth over the study period.', 'These results raise concerns about whether migrants in Europe are as likely as non-migrants to a
ge in good health.', 'We recommend that policies aiming to promote healthy ageing specifically address the health needs of the
migrant population, thereby distinguishing migrants from different backgrounds.', 'Electronic supplementary material The online
version of this article (10.1186/s12916-018-1044-4) contains supplementary material, which is available to authorized users.']
```

Figure 45: Qualitative error analysis 2.

3.4 Discussion, Comments/Notes and Future Work

During the implementation of the project there were many points during the implementation procedure where we wondered whether better approaches could be implemented in order for better results to be achieved. Furthermore, there were sometimes issues which could not be fixed. All these would be further discussed and analysed.

To begin with, the processing of the text could be greatly improved so that either the Fasttext model or the embeddings used to create the graph had less unnecessary information and as a result the algorithms learned the substance better. A better approach to this would be to implement stemming and lemmatization processes as well as other more advanced word processing methods such as Bag of words or glove embeddings respectively. A better text editing might improve the results of both Fasttext and embeddings. Especially in the case of the Fasttext model with the best cleaning we might have seen more variability in the results during the validation test.

In addition, in terms of embeddings, they could be improved or better defined if we experimented more with the input of the SentenceTransformer model. In our case we used the parameter: 'paraphrase-MiniLM-L6-v2' as input and no further test was performed. In addition, another parameter that determined the distances between the embeddings and contributed to the creation of the similarity table is the type of distance we chose. We chose to walk using the Euclidean distance but did not experiment with other types of distances that would probably allow a better distribution of the similarities.

Moreover, in this section we could also comment on the hasty overfit that was applied in the step of the intuitive baseline. Logics like: 'rank the last sentences as a claim' no matter what they contain are sure to have biased results, but nevertheless were implemented as they served the purposes of the first step.

Last but not least, regarding thoughts for future work, in addition to better text editing it would be very interesting to apply version 2 of the Fasttext approach to consider its own effects. Version 2 contains the use of pre-trained context vectors which we could not find on the internet for medical issues which concern us.

4.1 Members/Roles

Team “Abalysts” who consists of Nikos Kyriakakis and George Papadomarkakis who have so far collaborated on several if not all of the team assignments of AUEB’s postgraduate programme, Business Analytics. In this work, due to the small number of team members, there was a lot of flexibility in dealing with each aspect of the work. However, there was a distinction of sub-tasks as described below.

As for the part of the annotation, this was done under the orders of the project leader, Aris Fergadis, separately from each one and only in the end there was a form of cooperation - discussion in order to correct certain percentages of agreement. Then, regarding the intuitive baseline, the script in python was written by both members of the team. Everyone's responsibilities differentiated in the next steps as Nikos Kyriakakis took responsibility for the Fasttext approach (for both the argumentative and the structure label clustering) and George Papadomarkakis undertook the clustering through the Graph process. Finally, the report was written with the collaboration of both members of the team who of course had the main say in the points where they had worked the most.

4.2 Time Plan

The starting point of this project was a series of workshops where the project leader gave all the group instructions and material to take shape the original data sets. More specifically:

- 24/6/2021 – 1st Webinar:
    - Discuss annotation guide
    - Discuss annotator agreement
    - Get the first part of data to annotate
- 1/7/2021 – 2nd Webinar:
    - Answer questions, resolve problems, annotation agreement feedback
    - Get the second part of data to annotate
- 8/7/2021 – 3rd Webinar:
    - Group's project report
    - Present baselines
    - Get a sample dataset for our baselines / models
- 9/7/2021 – 22/7/2021:
    - Keep in touch for question or problems while annotating
- 23/7/2021:
    - Deadline for annotations delivery
- 30/7/2021:
    - Get the full annotated datasets
- 31/7/2021 – 6/8/2021:
    - Dealing with the theoretical background of the course in order to refresh our knowledge on the required areas of the project
- 7/8/2021 – 13/8/2021:
    - All datasets delivered by project leader, Aris Fergadis
    - Implementation of Intuitive baseline
- 17/8/2021 – 20/8/2021:
    - Implementation of Fasttext approach for both argumentative and structure labelling
- 20/8/2021:
    - Meeting with Mr Perrakis in order to get feedback for our work and get suggestions to improve what we had already done
- 21/8/2021 – 24/8/2021
    - Implementation of abstracts' clustering through the Graph method
- 25/8/2021:
    - Meeting with Mr Perrakis in order to get comments for the whole overview of the project so far and get suggestions for further improvement
- 26/8/2021 – 28/8/2021:
    - Corrections and further experimentation on project parts for which we had doubts

- 28/8/2021 – 2/9/2021:
    - Compilation of the report

4.3 Bibliography

1. S.E. Toulmin, The Uses of Argument, Cambridge University Press, 1958.
2. I. Habernal and I. Gurevych, Argumentation mining in user-generated Web discourse, Computational Linguistics (2017)
3. https://towardsdatascience.com/introduction-to-natural-language-processing
4. https://www.analyticssteps.com/blogs/7-natural-language-processing-techniques-extracting-information
5. https://towardsdatascience.com/optimising-a-fasttext-model-for-better-accuracy
6. https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture
7. https://pypi.org/project/sentence-transformers/
8. https://networkx.org

4.4 Appendix

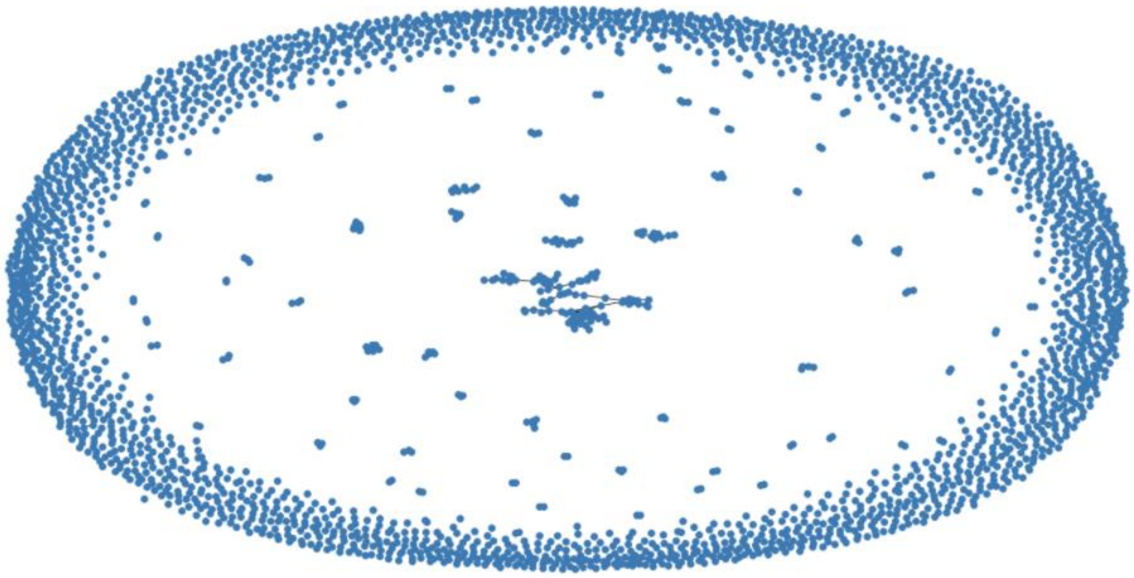


Figure 46: Graph plot (claim arguments)

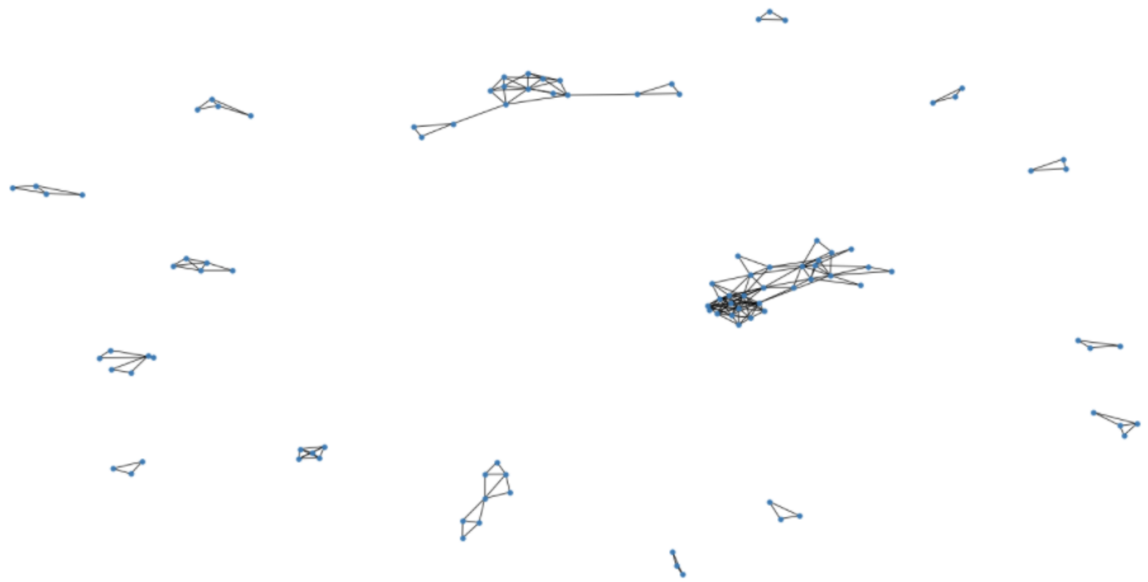


Figure 47: Communities with k=3 (similarity >0.6, claim arguments)

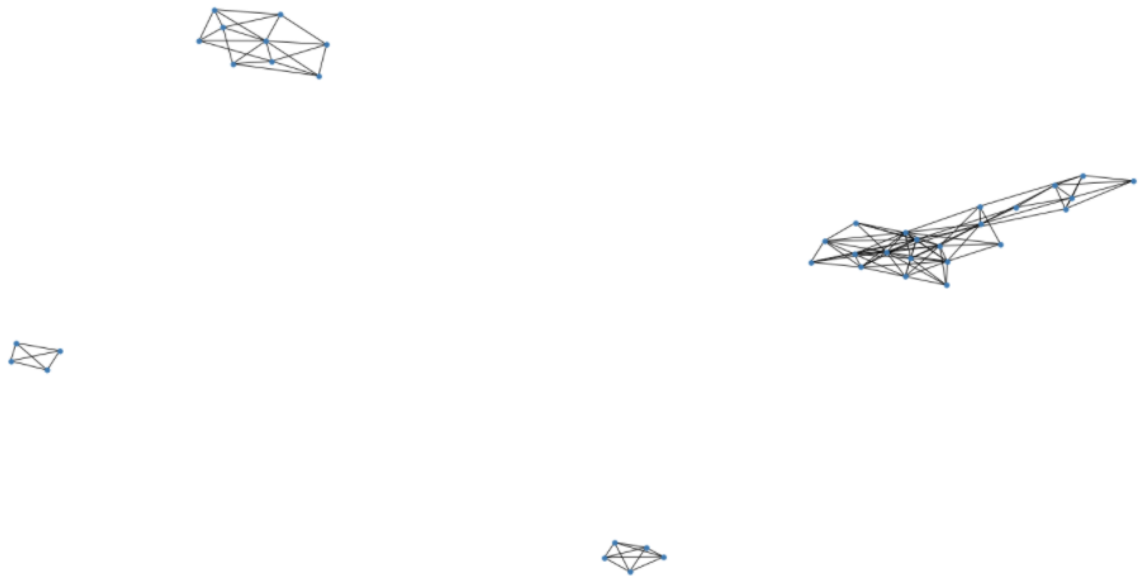


Figure 48: Communities with k=4 (similarity >0.6, claim arguments)

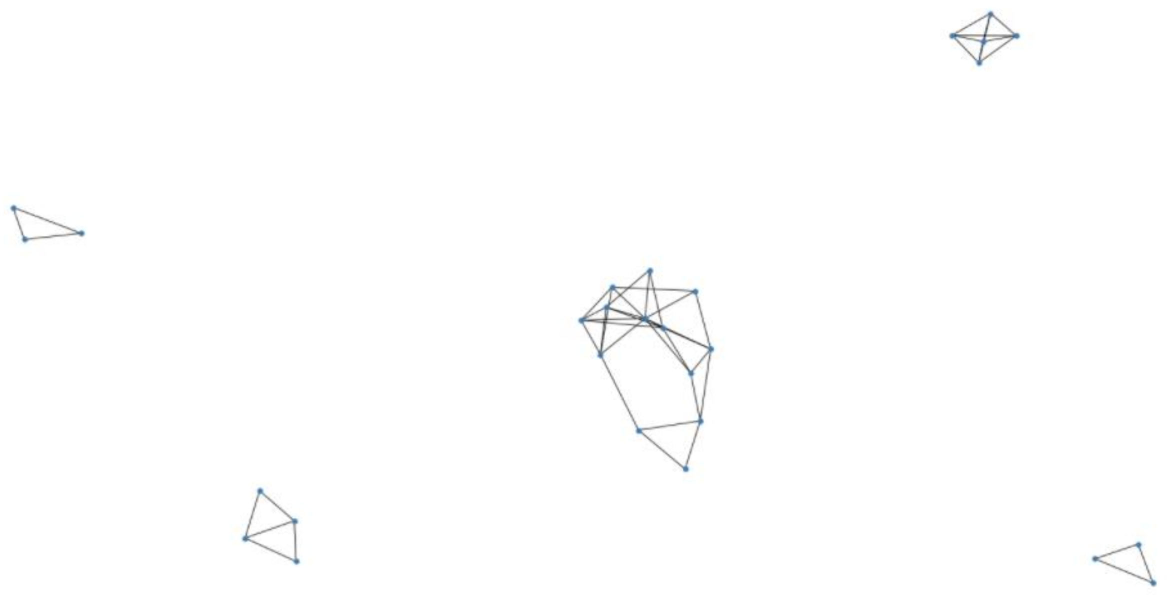


Figure 49: Communities with k=3 (similarity >0.65, claim arguments)

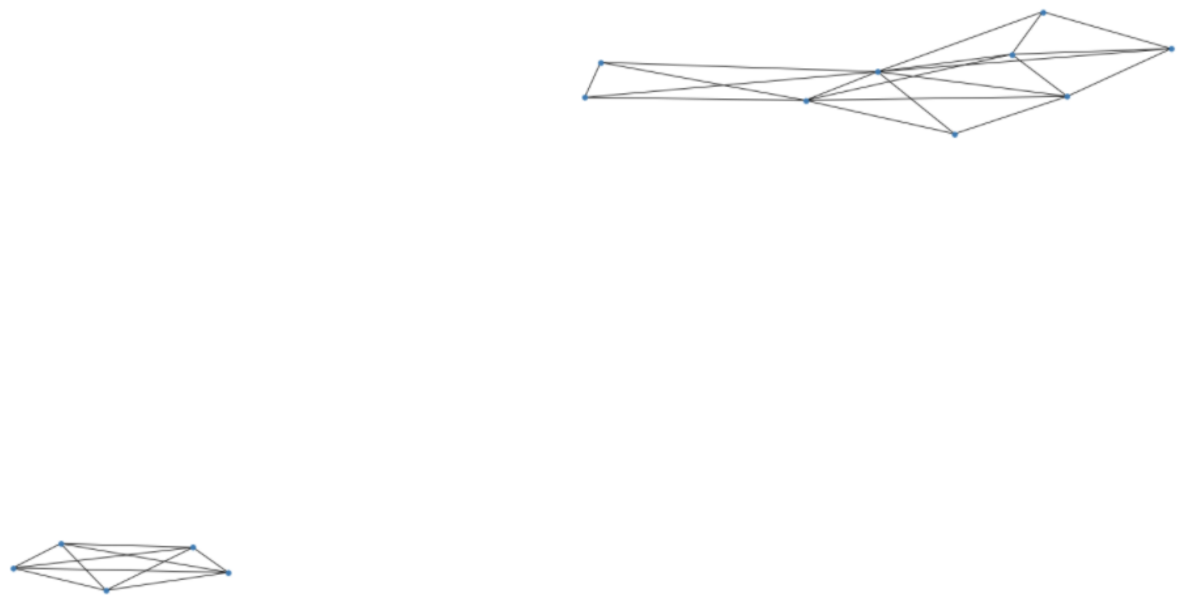


Figure 50: Communities with k=4 (similarity >0.65, claim arguments)

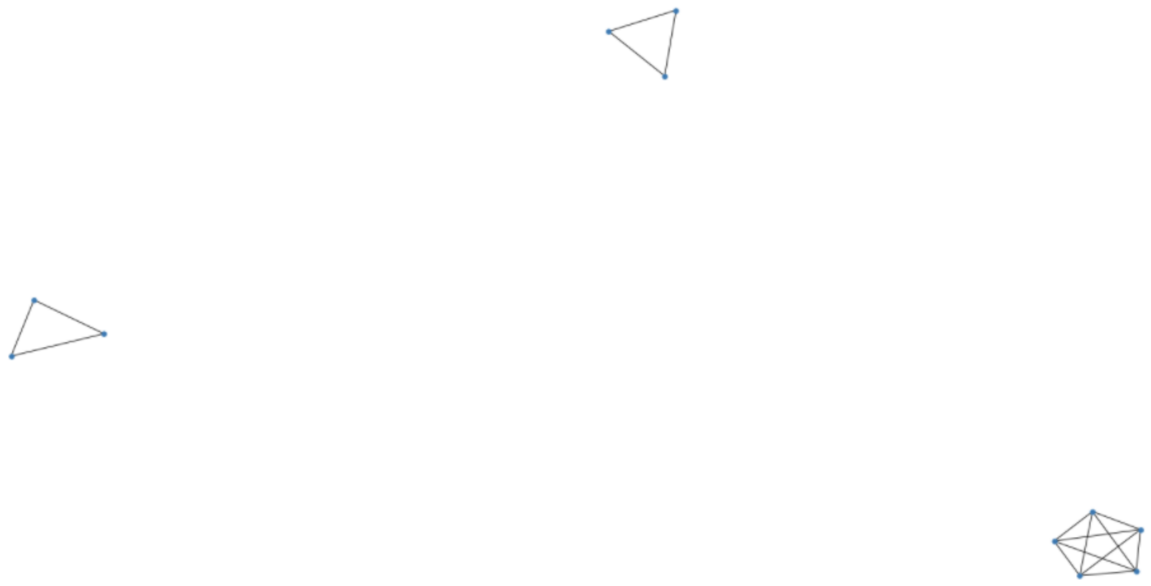


Figure 51: Communities with k=3 (similarity >0.7, claim arguments)