



**BUSINESS
ANALYTICS**
Master of Science



School of Management Science and Technology

M.Sc. in Business Analytics (Part - Time)
Machine Learning & Content Analytics
Pilot Project

Academic Year 2020-2021

Georgios Mantzos (p2822012)

Ioannis Dimitriou (p2822006)

Athens, August 2021

Contents

Abstract	3
1. General Overview of the Datasets	3
a. Extract – Transform – Load (ETL).....	4
1.2 Split Dataset into Train/Test/Validation sets	6
1.3 Exploratory Data Analysis on the Training Set	6
1.4 Building Lexicons	9
1.5 Classification based on the Baseline Intuitive model	11
2. Argument -Structure Prediction	12
a) Argument Prediction - Intuitive Baseline	12
i) Data Preparation for Model Fitting	12
ii) Train and Evaluate the Model - Get Predictions	13
b) Argument Prediction	14
i) Data Preparation for Model Fitting	14
ii) Train and Evaluate the Model - Get Predictions	15
c) Structure Prediction	16
i) Data Preparation for Model Fitting	16
ii) Train and Evaluate the Model - Get Predictions	17
d) Citance Prediction	18
i) Data Preparation for Model Fitting	18
ii) Train and Evaluate the Model - Get Predictions	19
3. K-Means Clustering.....	20
a. Data Preparation	20
b. Clustering Models.....	21

Abstract

The main purpose of this project is to use the data that were produced after the labeling of the abstracts by different documents for implementing Natural Language Processing models. This labeling refers to the arguments and structure of each document and the citations that they had in other documents.

This data is used in order to create argument, structure labeling, for each sentence of the documents and citances labeling for each citation that refers to them. Transformers Networks with BERT pre-trained classification model was used for that purpose, utilizing different approaches of sentence pair classification.

In addition to that, data describing characteristics of each document were combined to the labeling one in order to implement K-Means Algorithms and decide the optimal number of clusters for the documents.

1. General Overview of the Datasets

Conducting machine learning concepts and projects, requires well-defined datasets and some general overview of their internal structure. Concerning the Argument labeling, we will make usage of the *sample* dataset we were given at the beginning of the project and the *dataset_aueb_argument_v3.json*.

Both datasets will be loaded individually and they will be merged when they have the appropriate form.

For analysis and prediction purposes, we split the documents into sentences and labels (keeping a constant index) with the option to unite them afterwards.

Let's see some key points of the datasets that will be merged into one dataframe:

Datasets Strats
Dataset 1 length: 1017 abstracts
Dataset 2 length: 1669 abstracts
Dataset 1 Minimum Document ID: 0
Dataset 1 Maximum Document ID: 1016
Dataset 2 Minimum Document ID: 1017
Dataset 2 Maximum Document ID: 2685

Also the labels datasets have a slight difference which should be corrected before merging the datasets into one. The first label dataset has as labels **"CLAIM"**, **"EVIDENCE"**, **"NEITHER"**. The second label dataset has as labels **"CLAIM"**, **"EVIDENCE"**, **"NONE"**. So, we are going to convert the **"NONE"** label to **"NEITHER"**.

	Dataset A	Dataset B	Both Datasets
CLAIM	1048	1450	3419
EVIDENCE	1700	2260	6210
NEITHER	7827	14548	22375

a. Extract – Transform – Load (ETL)

Having combined the two source datasets, we need to make some data cleansing and transformation in order to conclude in our final dataset, which will be split in train/test/validation datasets.

First, we know that the first sentence of each Document is their title. It can produce some bias to our model, as it will always have the label '**NEITHER**'. So, there will be 2585 such labels more than they actually exist in the dataset. In order to do so, we will add 3 more columns in both dataframes:

1. **Sentences_Per_Doc**: Showing the number of sentences in each Document.
2. **Sentences_Id**: The real index of each Sentence.
3. **In_Doc_Sentence_Id**: The internal Id for any sentence in each Document, taking values from 0 to the "Sentences_Per_Doc" value of each document.

Having removed the title from the sentences data-frame and the corresponding rows in the labels data-frame, we end up to the following format of sentences/label dataframes:

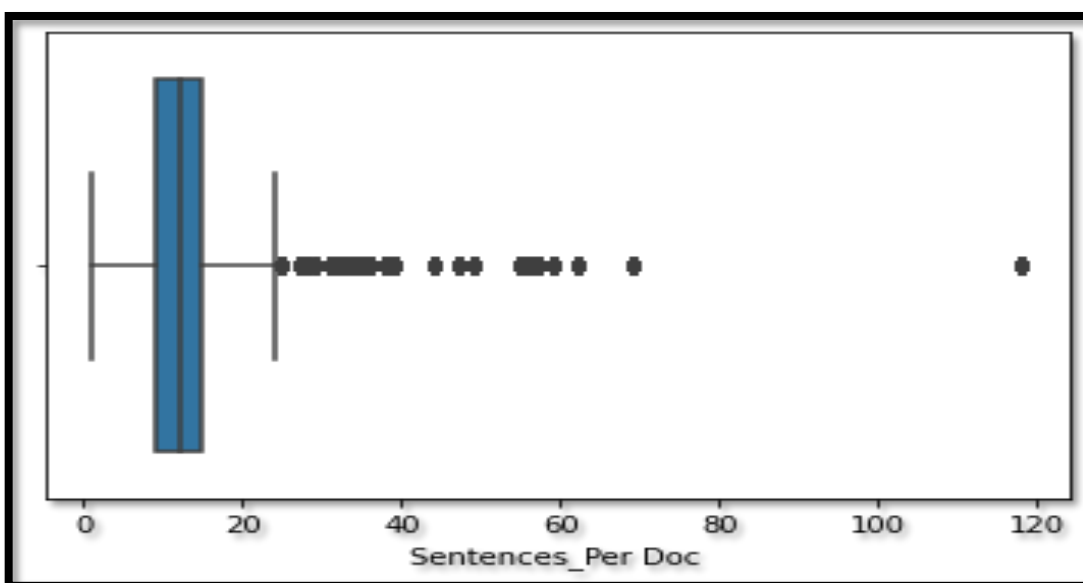
	doc_id	sentence	Sentences_Per Doc	Sentence_Id	In_Doc_Sentence_ID
0	0	Importance Visual assessment of amyloid positr...	16	1	1
1	0	Several immunoassays have been developed to me...	16	2	2
2	0	The agreement between CSF A β 42 measures from d...	16	3	3
3	0	Objective To determine the concordance between...	16	4	4
4	0	Design, Setting, and Participants The study in...	16	5	5

	doc_id	label	Sentences_Per Doc	Sentence_Id	In_Doc_Sentence_ID
0	0	NEITHER	16	1	1
1	0	NEITHER	16	2	2
2	0	NEITHER	16	3	3
3	0	NEITHER	16	4	4
4	0	NEITHER	16	5	5

Before splitting out our data to training and test set, we are going to explore our data and see if there are extreme values in how much sentences a document has.

Extreme outliers will probably conclude to bias in our argument labeling, as they will have much more 'NEITHER' labels than the majority and may also have much more of the rest labels.

But let's explore the distribution of that variable firstly. From the simple boxplot above, it can be inferred that there is, for sure, an extreme outlier of 118 sentences.



As it can be inferred from the boxplot above, there are some extreme outliers while the vast majority of the distribution is restricted in a range about 10-18.

So, let's explore some thresholds that will provide a more solid distribution of the number of sentences per abstract, but without losing much information.

Number of Abstracts with more than 20 sentences	66 (2,46% of total)
Number of Abstracts with more than 25 sentences	31 (1,15% of total)
Number of Abstracts with more than 30 sentences	24 (0,89% of total)

Our choice is based on the boxplot and the findings above is to keep a **threshold of 25 sentences** per abstract.

This is because on the one hand we do not lose much information with dropping documents above that threshold (**1.08 %**), but on the other hand we do not obtain much more information with a higher threshold of 30 ($1.08\% - 0.81\% = \mathbf{0.27\%}$).

So, a threshold of 25 sentences seems reasonable based on our data.

1.2 Split Dataset into Train/Test/Validation sets

Having prepared our data for analysis, for data consistency reasons, the split will be made into the `doc_ids` and after that the corresponding sentences and labels will be introduced in Train, Validation and Test sets.

So, we are going to split the documents as a whole and after that, labels and sentences will be introduced, using the `doc_ids`.

In our models the X will be the sentences dataset, while the Y will be the one with the corresponding labels.

The datasets that were produced after the data cleansing procedures will, firstly, be split into:

1. **Train-Validation Set** (85% of the Dataset)
2. **Test Set** (15% of the dataset)

We are going to use only the "**sentence**" column of the sentences dataset and only the "**label**" column of the labels dataset.

Also, a further split of the **Train-Validation Set** will be made, dividing it into :

1. **Train Set** (80% of the Train-Validation Set)
2. **Validation Set** (20% of the Train-Validation Set)

At this point, we will conduct Exploratory Data Analysis (EDA) for the Training set and not for the Test and Validation sets since they should be unknown sets for our model.

1.3 Exploratory Data Analysis on the Training Set

The training set on which the EDA will be implemented, has the following format:

	label	doc_id	sentence	Sentences_Per Doc	Sentence_Id	In_Doc_Sentence_ID
0	NEITHER	0	Importance Visual assessment of amyloid positr...	16	1	1
1	NEITHER	0	Several immunoassays have been developed to me...	16	2	2
2	NEITHER	0	The agreement between CSF Aβ42 measures from d...	16	3	3
3	NEITHER	0	Objective To determine the concordance between...	16	4	4
4	NEITHER	0	Design, Setting, and Participants The study in...	16	5	5
...
19066	EVIDENCE	2685	No statistically significant difference in con...	14	31999	10
19067	NEITHER	2685	Latanoprost 0.005% once daily reduced IOP more...	14	32000	11
19068	CLAIM	2685	Latanoprost had no statistically or clinically...	14	32001	12
19069	CLAIM	2685	There was no difference in hyperemia between t...	14	32002	13
19070	CLAIM	2685	Both concentrations of latanoprost reduced IOP...	14	32003	14

19071 rows × 6 columns

At this point, we will create 3 new variables which count the number of labels in each abstract. These variables will be “claims”, “evidences” and “neithers”.

	label	doc_id	sentence	Sentences_Per Doc	Sentence_Id	In_Doc_Sentence_ID	claims	evidences	neithers
0	NEITHER	0	Importance Visual assessment of amyloid positr...	16	1	1	2	7	7
1	NEITHER	0	Several immunoassays have been developed to me...	16	2	2	2	7	7
2	NEITHER	0	The agreement between CSF Aβ42 measures from d...	16	3	3	2	7	7
3	NEITHER	0	Objective To determine the concordance between...	16	4	4	2	7	7
4	NEITHER	0	Design, Setting, and Participants The study in...	16	5	5	2	7	7
...
19066	EVIDENCE	2685	No statistically significant difference in con...	14	31999	10	3	4	7
19067	NEITHER	2685	Latanoprost 0.005% once daily reduced IOP more...	14	32000	11	3	4	7
19068	CLAIM	2685	Latanoprost had no statistically or clinically...	14	32001	12	3	4	7
19069	CLAIM	2685	There was no difference in hyperemia between t...	14	32002	13	3	4	7
19070	CLAIM	2685	Both concentrations of latanoprost reduced IOP...	14	32003	14	3	4	7

19071 rows × 9 columns

Based on these new variables, we can have an insight look at the statistics of the labels.

In the table below, we can see some key strats of the arguments and have an enhancement about their behavior among the data, per abstract.

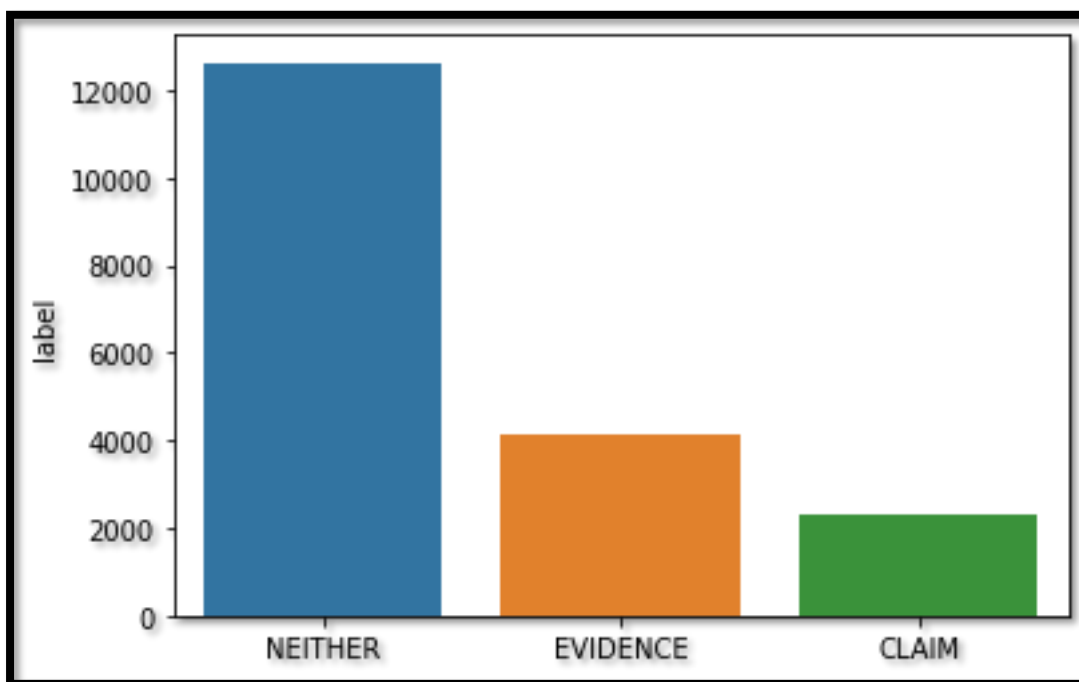
Argument	Min.	Max.	Avg.	Median
CLAIM	0	5	1.38	1.00
EVIDENCE	0	10	2.60	2.00
NEITHER	8	24	8.05	7.00

A first logical insight from the statistics for the baseline intuitive model can be that we want all the abstracts to have:

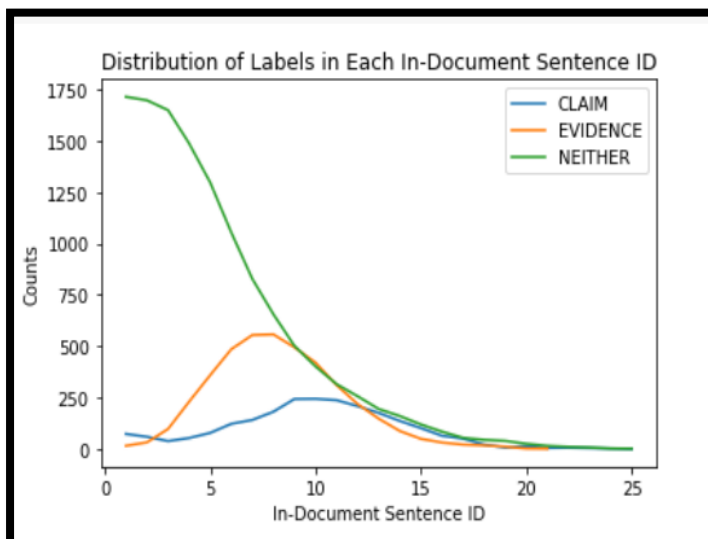
1. One up to 2 Claims
2. One up to 4 Evidences
3. Leave the rest sentences as neither.

In the next pages, we will produce and present some more insightful bar-charts and line-charts about the behavior of the labels and some very interesting plots concerning the most common words that we crossed from our analysis in the sentences.

The below bar-chart depicts, in a more insightful way, the distribution of the labels among the abstracts.



We can also see the distribution of each label in each In-Document Sentence ID:

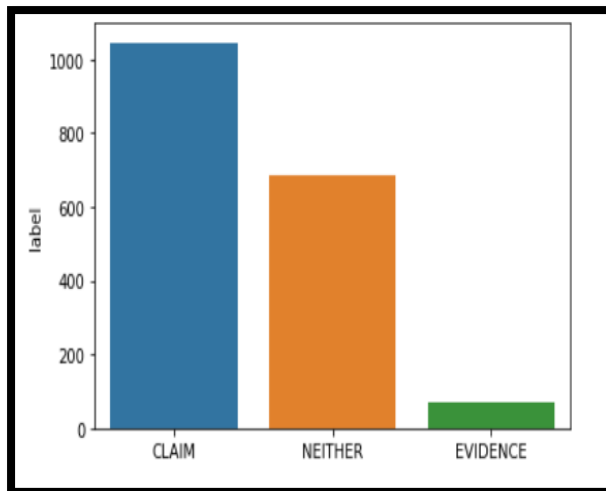


It is obvious that when an abstract comes up to its end, then a convergence among the three labels it is occurred (from 12th to 25th sentence)

Now, a new variable will be introduced, which will be a flag taking values:

- 1 if a sentence is the last one of the abstract.
- 0 if it is not.

In the following bar-chart we can see the distribution of the labels concerning the **last sentence** of the abstract:



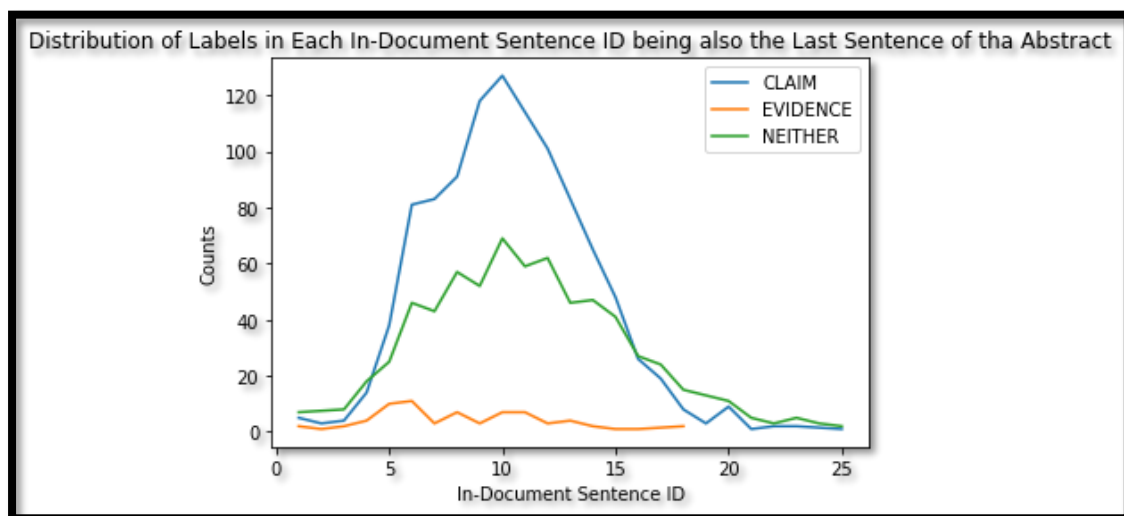
It is found that the 57.98% of the last sentences in each document are CLAIMS and the 45.74% of the CLAIMS are in the last sentence.

Additionally, the 3.88 % of the last sentences in each Document are EVIDENCE and the 1.68 % of the EVIDENCE are in the last sentence.

So, there is quite high probability for the last sentence of each document to be a CLAIM.

Moving on with our analysis, we concentrate in the interval of sentences that present high density of each label.

It is found that the 90.73 % of CLAIMS in the last sentence are in a range of [5th,16th] sentence of an abstract.



Having said that, we can set a new rule that governs the abstracts:

- If a sentence is the last one of an abstract and it is in the range [5,15] as for the In-Document Sentence ID, it is labeled as "CLAIM".

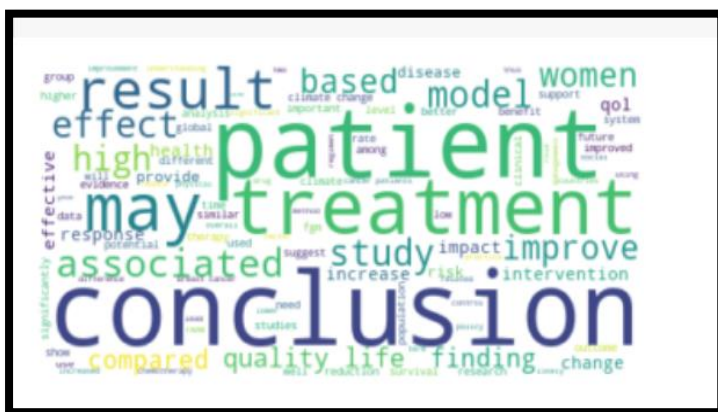
1.4 Building Lexicons

Before any tries of classification models, there is one more pending requirement. We need to build lexicon for the most common words of each label.

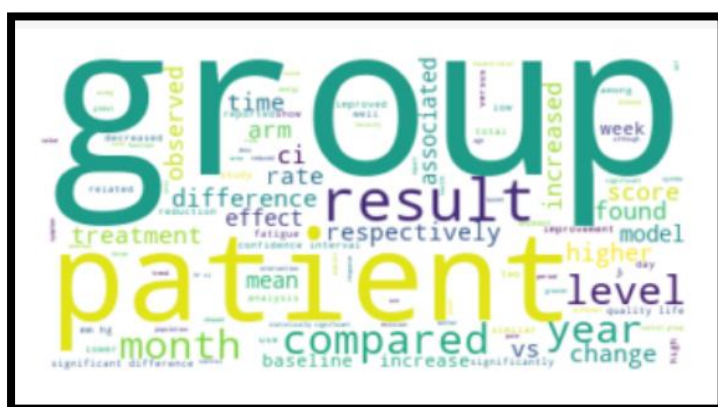
Firstly, we are going to define a **counter_vectorizer** which will transform the sentences to binary vectors. The **stop-words** of the English Language will be excluded. Also, we define a function in order to eliminate the numbers. We set lowercase parameter to True so a transformation of the sentences to lowercase is needed. The length of the vectors will be 20,000 which means 20,000 different words. That size is far from enough for the size of our data. This procedure will be followed for all the 3 labels separately, in order to build 3 characteristic lexicons.

Before fitting the Counter Vectorizer, let's have a visualization of each word that the sentences of each label are consisted of.

A **WordCloud** plot would be ideal to have the "big image".



From the WordCloud plot concerning the **CLAIMS**, we can observe "conclusion", "result" and "associated" as some of the most common words.



Respectively for **EVIDENCE**, some of the most common words are "patient", "group", "difference". Note that there are some common top words between **CLAIMS** and **EVIDENCE** but it seems logical since our data are based on scientific papers.



For NEITHER label, there is a plethora of top words and the majority of them is common with the ones being mentioned for CLAIMS and EVIDENCE.

The next step is to fit the counter_vectorizer and depict the number of the top-100 words from lexicons of each label that appear in each sentence of the dataframe. These are the following variables:

- 'claims_lexicon_counts'
- 'evidences_lexicon_counts'
- 'neithers_lexicon_counts'

These 3 new variables will be added to the master dataframe that will be used for the classification rule we are going to apply.

1.5 Classification based on the Baseline Intuitive model

Having done all that data preparation and transformation, it is time to review the findings and imprint a satisfactory classification rule. Based on our analysis, we end up to the following rule(s):

1. If it the last sentence and that sentence is between [5th.16th] interval, then the output label of the rule is CLAIM.
2. If the number of claims lexicons is greater than 1 and greater than the number of evidence and neithers lexicons respectively, and it is the pre last or second before the last sentence, then the outcome of the rule is CLAIM.
3. If the number of evidence lexicon is greater than the number of neither and claim lexicons respectively, and the number of evidence lexicon is greater than 3 and the output is not already claim, then the classification output is EVIDENCE.
4. All the other cases are considered as NEITHER

2. Argument -Structure Prediction

a) Argument Prediction - Intuitive Baseline

i) Data Preparation for Model Fitting

The approach chosen by our team is the **Transformers Networks Approach** and more specifically the second version that was proposed. This is the sentence pair classification task using BERT pre-trained model. Based on it, we need to make a formulation of our data in the form of **Sentence 1 – Sentence 2 – Label**. So, it is a fact that every model needs a specific data formulation which should be prepared before fitting the model.

First of all, in this task we have more data than the rest tasks of predictions as two datasets were delivered to us and they were merged in order have one bigger dataset. This way we conclude to a form that has for each sentence of train, validation and test set the following variables: “doc_id”, “sentence”, “label”, “Sentences_Per_Doc”. The Approach that will be followed is to make combinations of the Title of each document and each their Sentence successively. The first and the last column were used in order to create a new column having the Title of the Document that each Sentence belongs to and then they were dropped. Finally, there are 3 columns in the data frames if the 3 sets, “**sentence1**” (title), “**sentence2**” (the actual sentence), “**label**”.

Below, an image of the train set is provided.

Proceeding in the data preparation procedure, the labels are encoded as indicated below:

- 0 for “CLAIM”
- 1 for “EVIDENCE”
- 2 for “NEITHER”

What is more, a data frame is not proper for training such a model, so we formulate a dictionary that has 3 sub-dictionaries one for each set, train, validation and test. Below, a general image of this data set and an element of the train dictionary are provided.

```
{
  'test': Dataset({
    features: ['sentence1', 'sentence2', 'label', '__index_level_0__'],
    num_rows: 4112
  }),
  'train': Dataset({
    features: ['sentence1', 'sentence2', 'label', '__index_level_0__'],
    num_rows: 19071
  }),
  'validation': Dataset({
    features: ['sentence1', 'sentence2', 'label', '__index_level_0__'],
    num_rows: 4840
  })
}
```

```
{
  '__index_level_0__': 0,
  'label': 2,
  'sentence1': 'Concordance Between Different Amyloid Immunoassays and Visual Amyloid Positron Emission Tomographic Assessment',
  'sentence2': 'Importance Visual assessment of amyloid positron emission tomographic (PET) images has been approved by regulatory authorities for'
}
```

This “DatasetDict” object, as it is defined in python, is the raw form of the data, having the whole titles and sentences. We need to convert the text to numbers, which is the form that

the model can make sense of. This is done with a tokenizer. The two sequences need to be handled as a pair, and apply the appropriate preprocessing. Fortunately, the tokenizer can also take a pair of sequences and prepare it the way our BERT model expects. After tokenization, this is the image of our DatasetDict.

```
Dataset({
  features: ['__index_level_0__', 'attention_mask', 'input_ids', 'label', 'sentence1', 'sentence2', 'token_type_ids'],
  num_rows: 3207
})
```

Three new columns can be noticed and they are used in order to map each word of the sentences to the corresponding id of the vocabulary used for the BERT pre-trained model tokenizer. So, they will be used in training the model along with the label.

In PyTorch, the function that is responsible for putting together samples inside a batch is called a collate function. It's an argument you can pass when you build a DataLoader, the default being a function that will just convert your samples to PyTorch tensors and concatenate them. This won't be possible in our case since the inputs we have won't all be of the same size. We have deliberately postponed the padding, to only apply it as necessary on each batch and avoid having over-long inputs with a lot of padding. This will be achieved with the function **DataCollatorWithPadding** from the Transformers library.

After that, we define a **TrainingArguments** class that will contain all the hyperparameters the Trainer will use for training and evaluation. We have defined the "test-trainer" one. Now we are ready to define our model, which is the **AutoModelForSequenceClassification** using the BERT pre-trained model with 3 labels.

The final step is to define the Trainer of the model. To do so, we are going to use all the parameters that were defined until now and these are the following:

- **model**
- **training_args**
- **train_dataset = tokenized_datasets["train"],**
- **eval_dataset = tokenized_datasets["validation"],**
- **data_collator**
- **tokenizer**

ii) Train and Evaluate the Model - Get Predictions

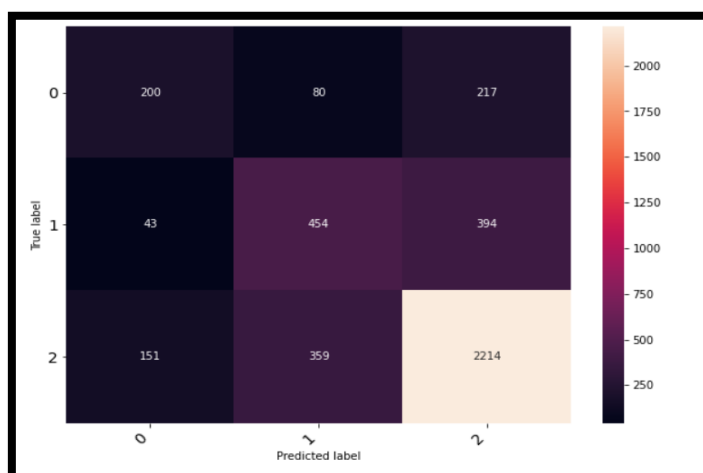
The model was trained and had 7152 total optimization steps and 3 epochs. The step 7000 achieved the lowest training loss with value of 0.239 which is an acceptable one. Below there is the evolution of the training loss while proceeding the steps.

[7152/7152 44:21, Epoch 3/3]

Step	Training Loss
500	0.701100
1000	0.599900
1500	0.587300
2000	0.538700
2500	0.508300
3000	0.415100
3500	0.410500
4000	0.419200
4500	0.406500
5000	0.339000
5500	0.265100
6000	0.248300
6500	0.258200
7000	0.239000

We can also see the classification report and the confusion matrix below.

	precision	recall	f1-score	support
0	0.51	0.40	0.45	497
1	0.51	0.51	0.51	891
2	0.78	0.81	0.80	2724
accuracy			0.70	4112
macro avg	0.60	0.57	0.59	4112
weighted avg	0.69	0.70	0.69	4112



b) Argument Prediction

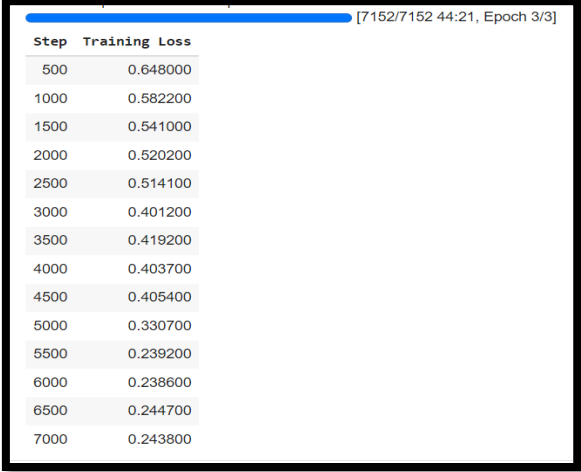
i) Data Preparation for Model Fitting

The next model that is going to be introduced and trained is again a model for argument prediction, but this one has the labels as they were distributed to us for the train set and not with classification rules as before. The whole procedure that was followed before for data preparation will again be followed for these data before training the model. So, it will not be described again. The only difference is that, as it was mentioned, the labels of the train set are

the actual labels from the dataset and they are not products of an intuitive classification rule as before.

ii) Train and Evaluate the Model - Get Predictions

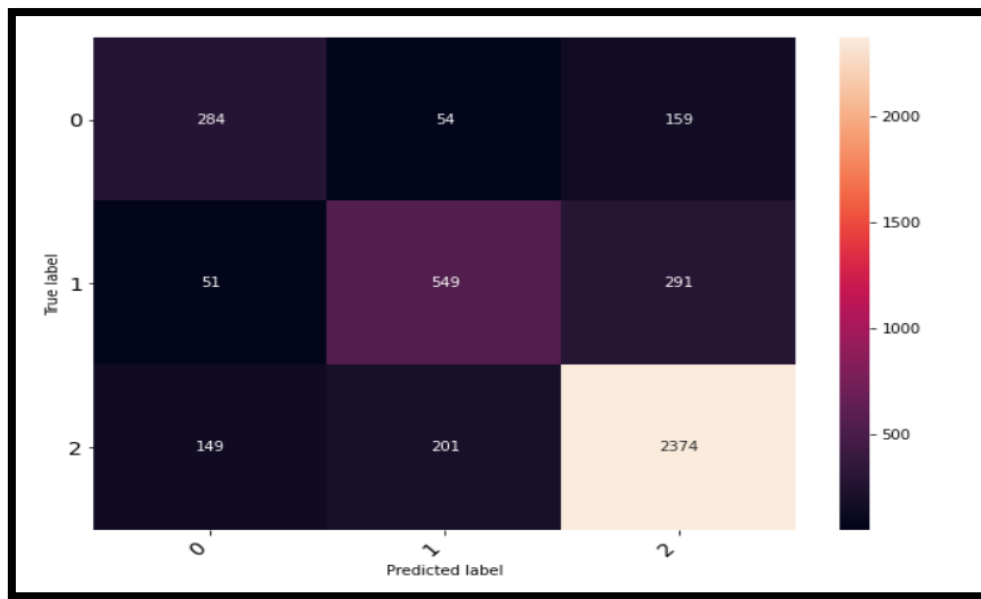
The model was trained and had 7152 total optimization steps and 3 epochs. The step 6000 achieved the lowest training loss with value of 0.2386 which is an acceptable one. Below there is the evolution of the training loss while proceeding the steps.



Step	Training Loss
500	0.648000
1000	0.582200
1500	0.541000
2000	0.520200
2500	0.514100
3000	0.401200
3500	0.419200
4000	0.403700
4500	0.405400
5000	0.330700
5500	0.239200
6000	0.238600
6500	0.244700
7000	0.243800

We can also see the classification report and the confusion matrix below.

	precision	recall	f1-score	support
0	0.59	0.57	0.58	497
1	0.68	0.62	0.65	891
2	0.84	0.87	0.86	2724
accuracy			0.78	4112
macro avg	0.70	0.69	0.69	4112
weighted avg	0.78	0.78	0.78	4112



c) Structure Prediction

i) Data Preparation for Model Fitting

For this classification task we have less data than before, coming from 951 Documents. A train-test-validation split was made, using the same proportions as mentioned in exploratory data analysis of the Argument Data. Below we can see a brief of summary of that split, indicating how many Sentences of how many Documents are the 3 sets consisted of. Note that There was an encoding issue in 3 documents and was resolved by removing them after communication with Mr. Fergadis.

Set	Number of Documents	Number of Sentences
Train	646	5824
Validation	162	1479
Test	143	1280

Since the approach that will be followed is again the title-sentence one, the titles were removed from the Sentences and a new column for them was created in order to have a pair of title-sentence for each sentence of our dataset. The initial labels are listed below:

- **RESULT**
- **BACKGROUND**
- **METHOD**
- **CONCLUSION**
- **OBJECTIVE**
- **NEITHER**

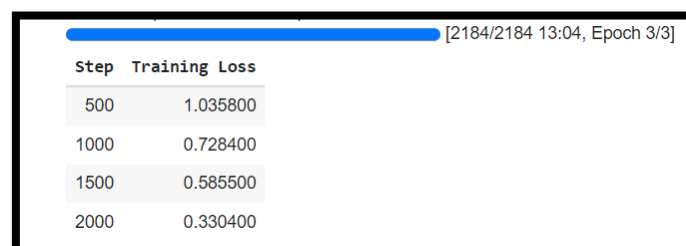
Note that the label “NEITHER” is not actually a structure label, but indicates the title sentence. Consequently, after removing the titles from the sentences there is no such label in our data.

The next step is to encode the rest 5 labels from 0 to 4 with the order that is listed above.

Proceeding, the same steps as in the previous models was followed in order to conclude to a data form that the BERT model needs to run. The only difference from the previous models, in the Train Arguments is that, now in the definition of the model we inserted the value of 5 as number of labels as it is logical.

ii) Train and Evaluate the Model - Get Predictions

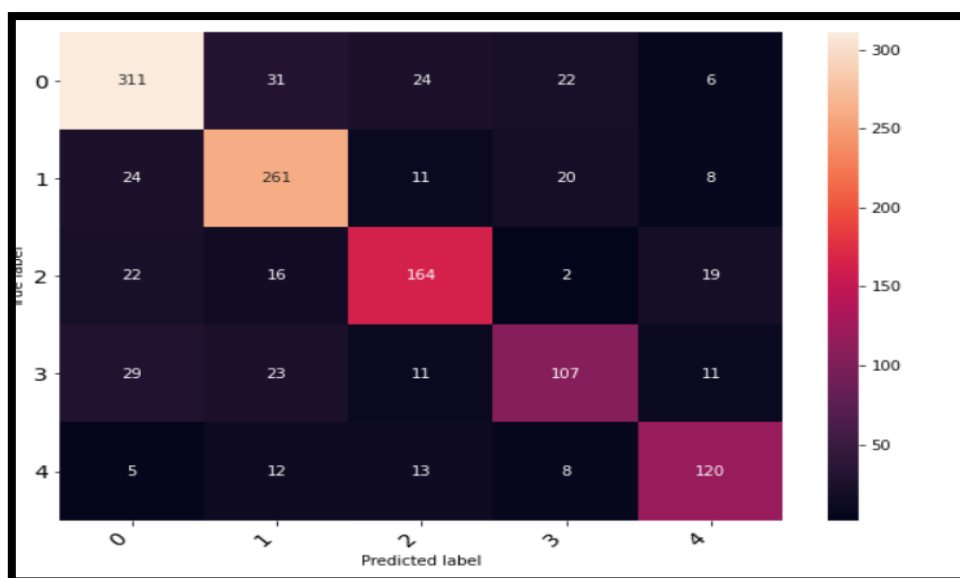
The model was trained and had 2184 total optimization steps and 3 epochs. The step 2000 achieved the lowest training loss with value of 0.3304 which is an acceptable one. Below there is the evolution of the training loss while proceeding the steps.



Step	Training Loss
500	1.035800
1000	0.728400
1500	0.585500
2000	0.330400

We can also see the classification report and the confusion matrix below.

	precision	recall	f1-score	support
0	0.80	0.79	0.79	394
1	0.76	0.81	0.78	324
2	0.74	0.74	0.74	223
3	0.67	0.59	0.63	181
4	0.73	0.76	0.75	158
accuracy			0.75	1280
macro avg	0.74	0.74	0.74	1280
weighted avg	0.75	0.75	0.75	1280



d) Citance Prediction

i) Data Preparation for Model Fitting

The last model was about Citance Prediction. The approach that was followed is again the same with pre-trained BERT model with the Sentence-Sentence approach, but this time we had the Claim – Sentence version. So, instead of having a column with the title of each document for each of their sentences, it was essential to create one having the claim of this Document, or all the claim sentences concatenated, if there is more than one and the sentence of the citance.

In order to use the claim labels of the structure labeling data, note that again the 3 documents that had the encoding issue were dropped. This ensured that we are going to have the same documents that also exist in the structure labeling data set.

The counts of the 3 sets (train-test-validation) are shown below:

Set	Number of Documents	Number of Sentences
Train	482	3207
Validation	109	634
Test	114	1322

That difference in the counts was derived from the fact that some documents from the structure dataset do not have claim sentences or some other documents from citances dataset do not have a citance sentence

Here the labels are different from before and they are the following:

- **NEUTRAL**
- **IRRELEVANT**
- **POSITIVE**
- **NEGATIVE**
- **NEITHER**

The last one is again the label that indicates the title sentence that was dropped. So, the 4 remaining labels were encoded from 0 to 3 with the order that are shown above.

What is more, the same procedure as in the 3 previous modes was followed again in order to make the essential transformations in the data and define the arguments for the Train of the model. The only difference is that in the model definition the number of labels was set to 4, which is the number of the labels of this data set.

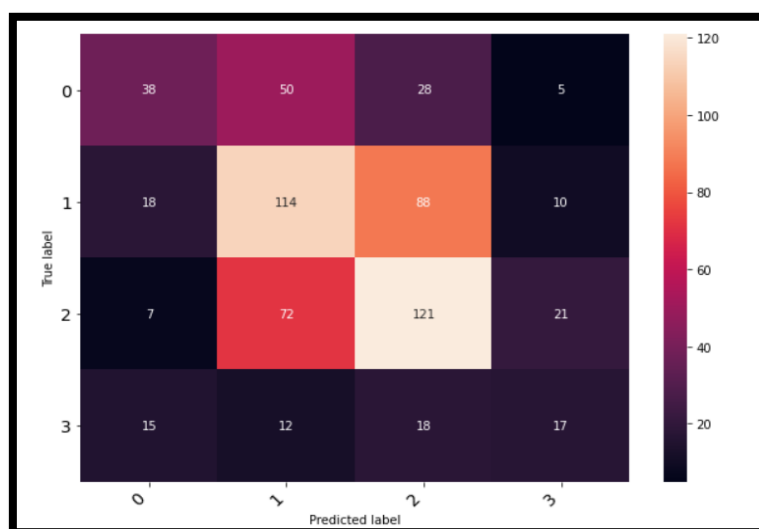
ii) Train and Evaluate the Model - Get Predictions

The model was trained and had 1203 total optimization steps and 3 epochs. The step 1000 achieved the lowest training loss with value of 0.491 which could be improved if more data were available. Below there is the evolution of the training loss while proceeding the steps.



We can also see the classification report and the confusion matrix below.

	precision	recall	f1-score	support
0	0.49	0.31	0.38	121
1	0.46	0.50	0.48	230
2	0.47	0.55	0.51	221
3	0.32	0.27	0.30	62
accuracy			0.46	634
macro avg	0.44	0.41	0.42	634
weighted avg	0.46	0.46	0.45	634



3. K-Means Clustering

a. Data Preparation

In this final task, the objective is to use the data that were used in the Structure label prediction task. The difference is that except from the “document”, “sentences” and “label”, two new objects will also be loaded by that json file, which are the following:

- **“project objective”**: The objective that each project that the documents belong to. It is in the form one or more sentences.
- **“eu call”**: It is an id indicating the EU Call project that the project was funded by. For this column there are also reference data that shows one or more sentences for each eu call id.

We need to use the whole abstract for clustering, so after the loading of the data, this is the task that takes place first. After that, the data frames that were created in the citances prediction task will be used, which are the ones having the claim sentence or sentences for each document concatenated and the claim(s) and evidence(s) of each document concatenated. These are merged with our data frame and form two new columns **“claim”** and **“claim_evidence”**. Note that documents with more than 26 or less than 4 sentences are eliminated from our dataset.

The next step is to merge the data frame with the reference data of the EU CALLS and take the text for each EU CALL.

Finally, the following columns are kept in the data frame and will be used for clustering models, **“abstract”**, **“project_objective”**, **“EU_CALL”**, **“claim”**, **“claim_evidence”**. All of them contain text.

However, text is not the appropriate form for clustering, so it is essential to create embeddings before fitting clustering models. In order to do that, a Keras Tokenizer was set up, containing the 15.000 most used words of our data. After that, the tokenizer is used to generate tokens which are used to transform text to sequences of numbers. Each one of these numbers corresponds to a word from the vocabulary created by the tokenizer.

To proceed, we now have to make sure that all text sequences we feed into the model have the same length. This was achieved with Keras pad sequences tool. It cuts of sequences that are too long and adds zeros to sequences that are too short. We chose to make all sequences 200 words long, a threshold which both conforms to the size of the text that each of our columns contain and it does not feed clustering models with massive size of data.

What is more, Glove Embeddings with dimension equal to 300 are downloaded and used in order to create the embeddings matrix and embeddings index that will be used for fitting the clustering models and see the optimal number of clusters for each combination of data for clustering.

The same procedure will be followed for all the possible combinations of the columns of the final data frame. They will be inserted together in a new embedding for each model. Below we are going to see all the possible combinations that were tested and point out some useful insights.

b. Clustering Models

The embeddings that can be created and combined are split in two categories:

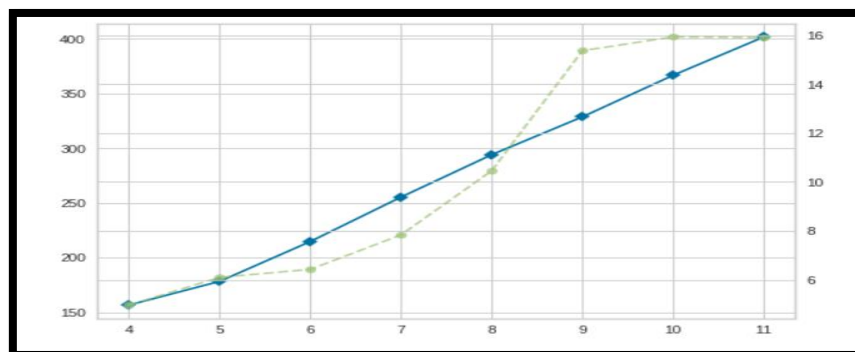
- Document embeddings for the
 - Abstract;
 - Project objective (each abstract belongs to a project)
 - EU Call (each project belongs to an EU call).
- Sentence embeddings for the argument.
 - Claim only.
 - Claim and Evidence.

Clusters were tested using:

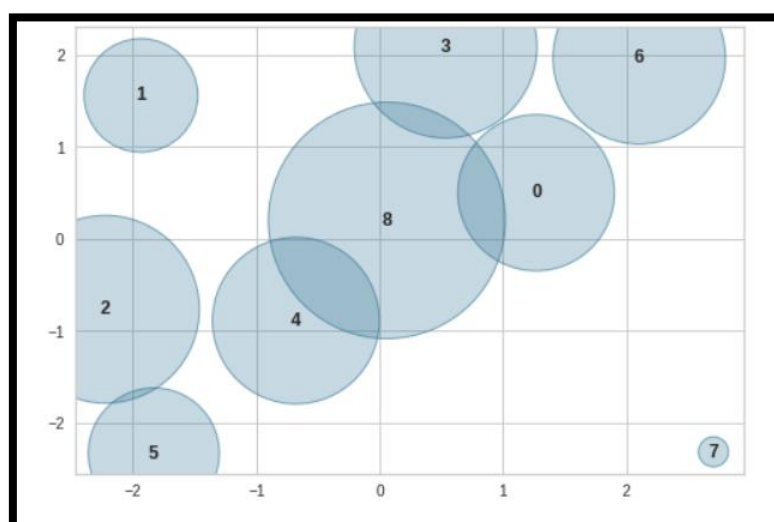
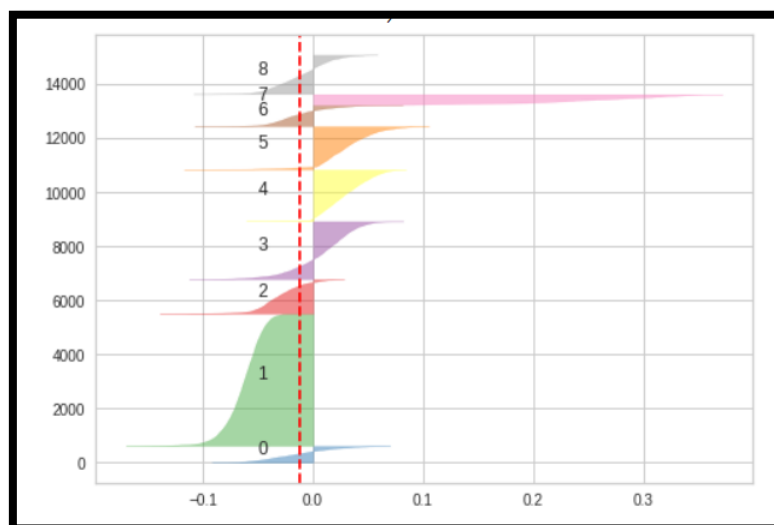
1. Only DE from the abstract.
2. Abstract DE and any combination of the other features (Project objective, EU Call).
 - a. Abstract
 - b. Abstract – EU Call
 - c. Abstract – Project Objective
 - d. Abstract – Project Objective – EU Call
3. Abstract DE and any combination of the features and the argument (claim/evidence) embeddings.
 - a. Abstract - Project Objective – Claim
 - b. Abstract - Project Objective - Claim/Evidence
 - c. Abstract - EU Call – Claim
 - d. Abstract - EU Call - Claim/Evidence
 - e. Abstract - Project Objective - EU Call – Claim
 - f. Abstract - Project Objective - EU Call - Claim/Evidence

The clustering results were not so satisfying for all the embeddings that were used. Below there is a description for the models that produced results that can be meaningful. For every model we are going to present a K-Elbow plot which will give directions to how many clusters should be examined for a specific model. In addition to that, Silhouette plots will be shown for the chosen number of clusters and finally an image of the clusters in a plot.

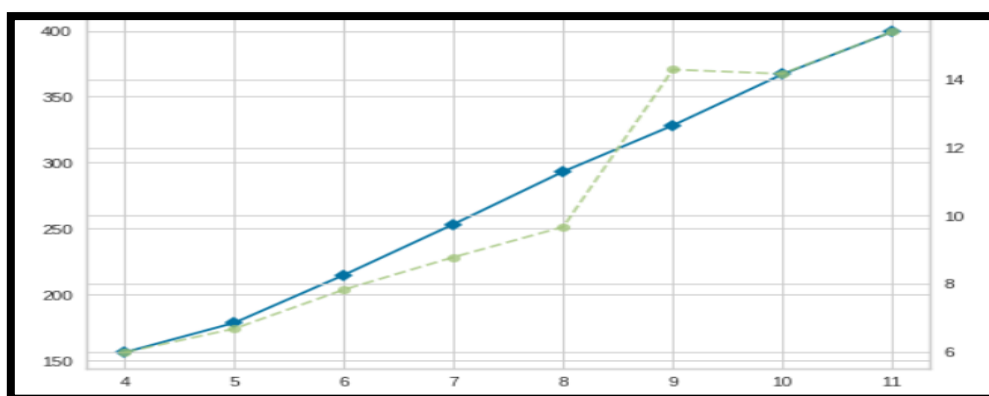
I. Abstract - EU Call Word Embeddings Clustering



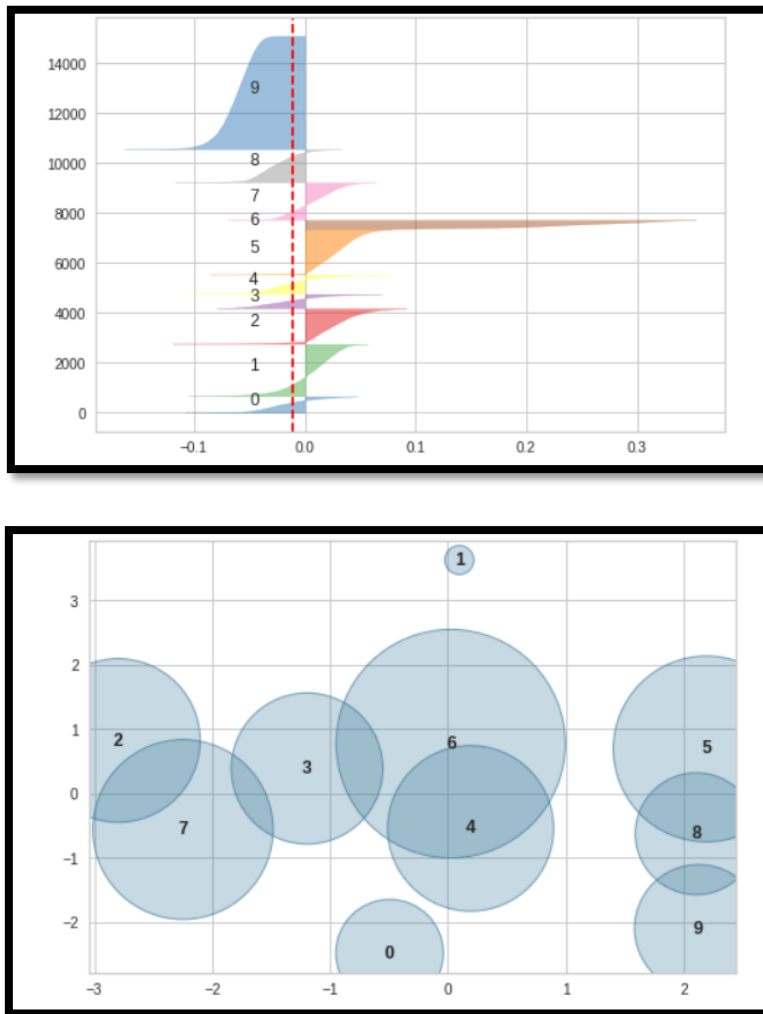
It can be inferred that the most meaningful number of clusters for that model is 9. So, let's see the Silhouette plot for 9 clusters and an image of them in a plot.



II. Abstract - Project Objective - EU Call - Claim/Evidence Word Embeddings Clustering



Also, in this case the most meaningful number of clusters seems to be 9. Below we can see the other two plots for that number of clusters in this model.



Concluding, the number of clusters for the best fit in both cases is equal to 9. What is more, this number was also the optimal for most of the rest models fitted, however these models were not satisfying. The Silhouette of the both the cases shown above is not the best possible and overlapping of the clusters seems to exist. The improvement of this could be the objective of further research in both the implementation of the clustering models and the creation of embeddings for them.