How to Disagree Project: Contradiction Detection

Laura Ciurca 942318
Università degli Studi di Milano
Data Science and Economics

Contents

List of Figures

Introduction

The aim of the project is to build a model able to predict contradictions in text, which has been identified as one of the 7 different types of arguments used in conversation and disputes according to Paul Graham in its essay. Contradiction detection is one of the biggest challenges in natural language processing considering especially the wide range of types of contradiction a conversation can face. But what it is a contradiction? Formally, a contradiction refers to a situation in which two sentences are really unlikely to be true together. Taking into consideration two sentences, a contradiction between them occurs when they cannot be both true. Contradictions can be revealed trough the presence of negation and antonymy as well as differences in the structure of the text or in the lexical context.

Another important concept which must be explained for the comprehension of this analysis is entailment. Entailment occurs when both premise an hypothesis are true. Formally, a sentence A is said to entail another sentence B if B is true and A can be logically derived from it.

In this project, two approaches have been implemented to the problem trying to recognize patterns in sentences. The first approach is based on extracting those features that could characterize contradictions in text, while the second one will make use of Word2Vec sentence encoders to find out if it can capture information in the sentences and lead to contradiction identification.

Four different machine learning models, more precisely Decision Tree, Random Forest, Sparse Tensor Classifier and ANN will be trained and tested for contradiction identification.

Research Question and Methodology

Detecting contradictions in text can be regarded as a classification problem. A contradiction occurs when a logical proposition is currently equal to its opposite, such as ”Today is raining and it is not”. Contradiction detection has been recently studied, especially after the so-called RTE (Recognizing Textual Entailment) challenge taken at the Text Analysis Conference in 2008, which tested its participants to develop a system able to determine whether the meaning of one sentence is entailed/inferred from another one, implying so the opposite situation, that is contradiction. In recent researches, there has been noticed the propensity of using Support Vector Machines to address the entailment task with different kinds of feature extraction, including syntactic and semantic features [2]. Many other researches have opted for the implementation of neural network architecures, such as LSTM, which are becoming more and more used for various learning problems related to sequential data.

In this project, Decision Tree, Random Forest, Sparse Tensor and ANN classifiers have been trained.

2.1 Decision Tree

A decision tree is a supervised predictive modelling approach used as a decision-making tool, since it allows to visualize and understand decision's options and possible outcomes. A tree predictor has the structure of an ordered and rooted tree T, whose internal nodes are assigned tests, functions mapping information to the children nodes, and whose leaves are assigned with labels. So, the decision tree structure starts from the root node, which is the highest one and is the global element. The top node originates all paths leading to leaf nodes, which contain questions to be answered.

2.2 Random Forest

Random Forest is a supervised learning algorithm, consisting in creating a forest made of a big number of decision trees that act as an ensemble learning method for classification and regression. Each single tree splits out a class prediction and the one that has the most votes becomes the model's prediction.

2.3 Sparse Tensor Classifier

Sparse Tensor Classifier (STC) is a supervised classification algorithm for categorical data inspired by the notion of superposition of states in quantum physics. Unlike most machine learning algorithms, STC results in high performance without the need of data pre-processing and parameter tuning, providing also a meaningful explanation of the classifier output.[3]

2.4 Artificial Neural Network

An artificial neural network (ANN) is a machine learning technique that is a brain-inspired system designed to simulate the way the human brain works as regards information analysis. ANN is characterized by hundreds or thousands of neurons, that are connected to each other by nodes and each of them consists of input and output units. An input unit receives information so that the neural network tries to learn the information to produce an output and to optimize it using learning rules called back-propagation.

Experimental Results

3.1 Dataset

The dataset used for learning the model in this analysis has been taken from Kaggle, named "Contradictory, My Dear Watson"[1], consisting into premise-hypothesis pairs in fifteen different languages, labeled by three categories: entailment (0), neutral(1) or contradiction(2). Below an example of the three categories according to the following premise:

"He came, he opened the door and I remember looking back and seeing the expression on his face, and I could tell that he was disappointed."

- Entailment:

  Just by the look on his face when he came through the door I just knew that he was let down.We know that this is true based on the information in the premise.

  We know that this is true based on the information in the premise.

- Neutral:

  He was trying not to make us feel guilty but we knew we had caused him trouble.

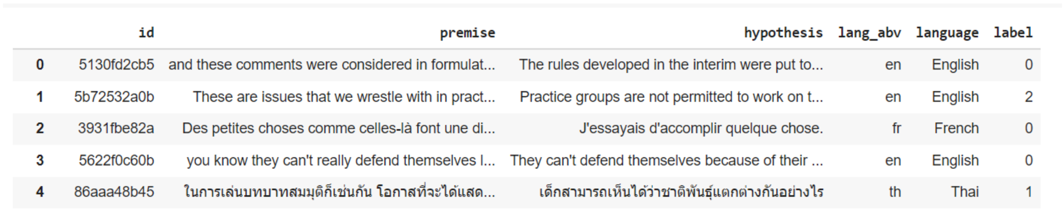
  This might be through but we cannot be sure considering the information in the premise.

- Contradiction:

  He was so excited and bursting with joy that he practically knocked the door off it's frame.

  We know this is not true, because it is the complete opposite of what the premise says.

The provided dataset is already split into training and test sets, both containing the ID, premise, hypothesis, the language of the text and its two-letter abbreviation. Training set contains 12120 rows, while test set 5195.



	id	premise	hypothesis	lang_abv	language	label
0	5130fd2cb5	and these comments were considered in formulat...	The rules developed in the interim were put to...	en	English	0
1	5b72532a0b	These are issues that we wrestle with in pract...	Practice groups are not permitted to work on t...	en	English	2
2	3931fbe82a	Des petites choses comme celles-là font une di...	J'essayais d'accomplir quelque chose.	fr	French	0
3	5622f0c60b	you know they can't really defend themselves l...	They can't defend themselves because of their ...	en	English	0
4	86aaa48b45	ในการเล่นบทบาทสมมุติก็เช่นกัน โอกาสที่จะได้แสด...	เด็กสามารถเห็นได้ว่าชาติพันธุ์แตกต่างกันอย่างไร	th	Thai	1

Figure 3.1: ”Contradictory, My Dear Watson” Training Set

3.2 Data Cleaning and Feature Extraction

The datasets have been imported directly on Google Colab through the Kaggle’s API. Just English sentences have been considered for this analysis, which enclose almost 60% of the initial datasets, dominating among all the languages. After this first filtering, training and test sets sizes are respectively 6870 and 2945 rows. Premise and hypothesis attributes have been taken into consideration for feature extraction.

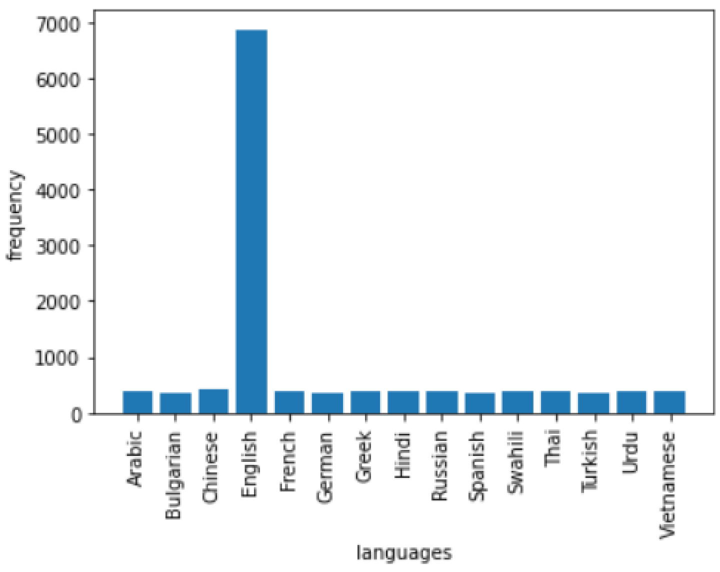


Figure 3.2: Sentences distribution according to Languages, Training Set

Fig.3.3, created through Word Cloud, is a visual representation of the most important keywords present in the hypothesis of the three categories referring respectively to entailment, neutral and contradiction. Words are displayed from small to large according to their frequency in sentences.

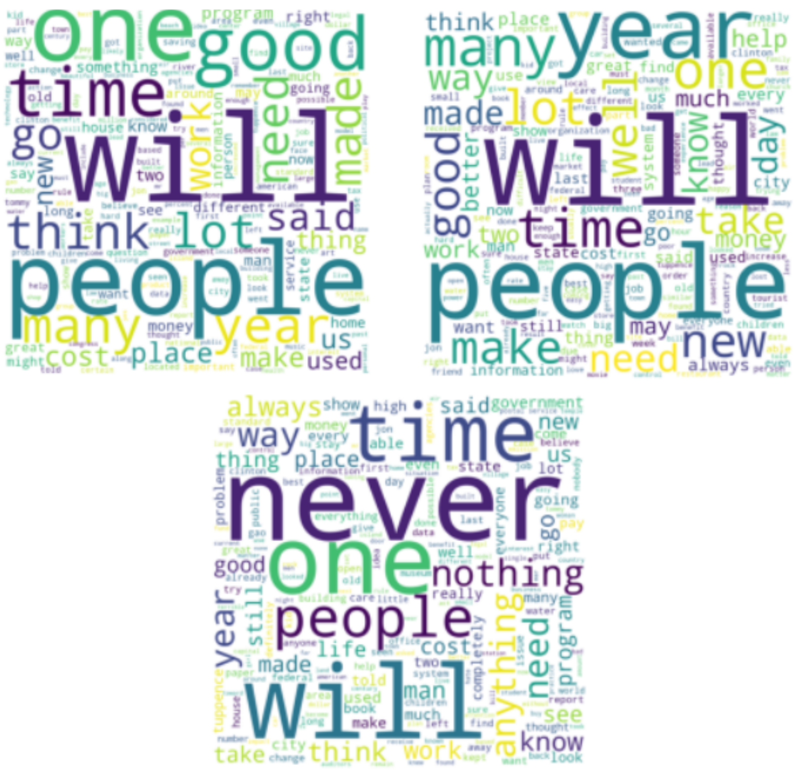


Figure 3.3: Wordcloud visualization of data according to entailment, neutral or contradiction categorization

In the training set, there are 6870 records, which contain

- 2427 records classified as entailment
- 2277 records classified as neutral
- 2166 records classified as contradiction

Therefore, there is no class imbalance.

Also word distribution has been checked on averaging the length of sentences in premise and hypothesis texts:

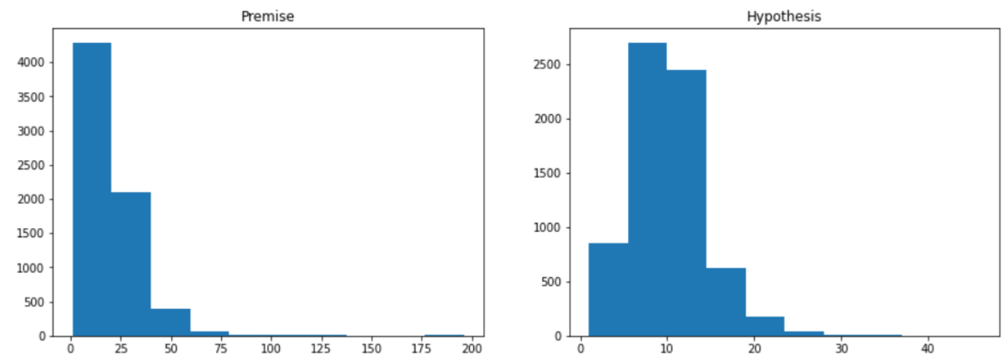


Figure 3.4: Sentences distribution according to word count, Training Set

First step taken in pre-processing consists in cleaning the sentences referring to premise and hypothesis of training and test sets, including lower case conversion and double space and punctuation removal. Stop word removals have not been considered as well as word stemming, considering that words like ”not” are fundamental for identifying contradiction between two sentences. After having defined a vocabulary of contractions, revealed contractions have been expanded to full words in such a way to allow better contradiction identification, especially as concerns antonym detection.

After the cleaning process, different features have been extracted:

- jaccard similarity
- cosine similarity
- polarity
- sentence similarity
- negation
- antonyms
- numeric mistmatch
- subject-subject similarity

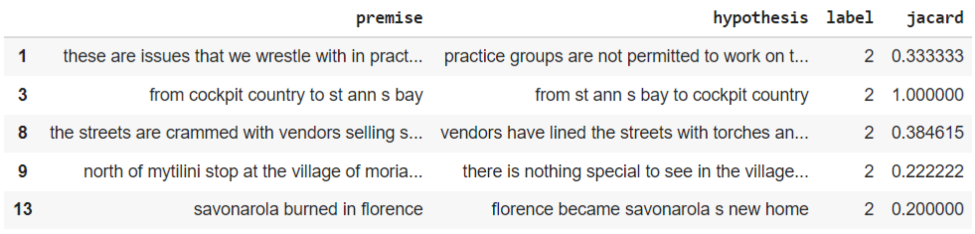
3.2.1 Jaccard similarity

One of the extracted features is Jaccard similarity which could be useful for contradiction detection. It turns to be 0 when the two sentences are completely different and 1 when they are equal. It measures the amount of similarity between premise and hypothesis texts, allowing to detect contradiction. Formally, it consists of the intersection size of two sets, A and B, divided by the size of their union, defined by:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

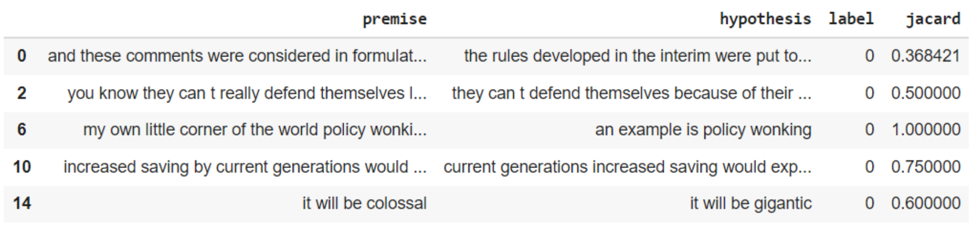
Jaccard similarity compares the boolean representation of premise and hypothesis, meaning that it is considering just the presence or absence of words without counting how many times they appear. In this way, by capturing the relation between the amount of similarity between the two sentences, it is possible to reveal if they are contradicting.

By checking the computed Jaccard similarity on the training set, it is possible to notice that sentences labeled as contradiction present a low similarity as per Fig.3.5, while the ones defined as entailment are characterized by a high similarity as represented in Fig.3.6.



	premise	hypothesis	label	jacard
1	these are issues that we wrestle with in pract...	practice groups are not permitted to work on t...	2	0.333333
3	from cockpit country to st ann s bay	from st ann s bay to cockpit country	2	1.000000
8	the streets are crammed with vendors selling s...	vendors have lined the streets with torches an...	2	0.384615
9	north of mytilini stop at the village of moria...	there is nothing special to see in the village...	2	0.222222
13	savonarola burned in florence	florence became savonarola s new home	2	0.200000

Figure 3.5: Jaccard similarity for contradiction



	premise	hypothesis	label	jacard
0	and these comments were considered in formulat...	the rules developed in the interim were put to...	0	0.368421
2	you know they can t really defend themselves l...	they can t defend themselves because of their ...	0	0.500000
6	my own little corner of the world policy wonki...	an example is policy wonking	0	1.000000
10	increased saving by current generations would ...	current generations increased saving would exp...	0	0.750000
14	it will be colossal	it will be gigantic	0	0.600000

Figure 3.6: Jaccard similarity for entailment

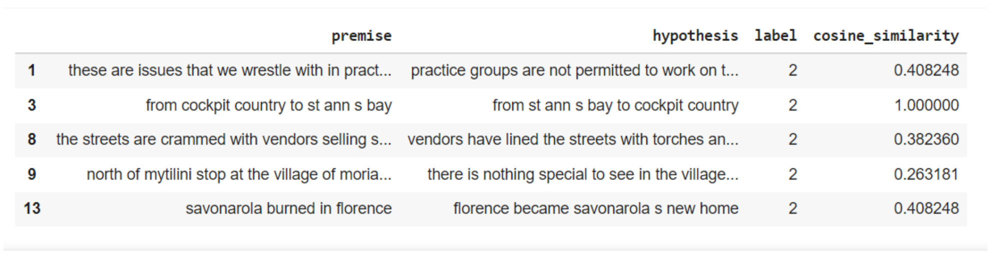
3.2.2 Cosine similarity

Also cosine similarity has been performed on premise and hypothesis sentences. While Jaccard similarity takes into consideration only unique sets of words for each sentence, cosine similarity considers the total length of the vectors. Both measures have been considered, since there can be sentences in which duplication does not matter as well as sentences in which it could matter to identify contradiction. After having created a vector of counts of words in sentences, cosine similarity can output a number between 0 and 1. Higher is the output, higher is the similarity between the two sentences. Mathematically, cosine similarity is given by the dot product and magnitude of the two vectors of attributes, A and B, as follows:

$$similarity = \cos(\theta)\frac{A \cdot B}{|A| \cdot |B|} \tag{3.2}$$

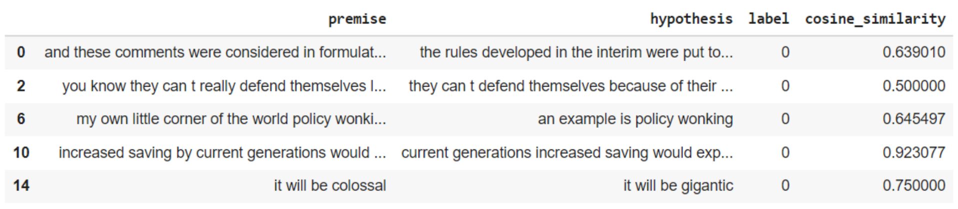
This formula has been applied after having turned the text into a vector by simply counting the words in the pairs of sentences, so that each vector contains a counter for each word, ignoring word order. Higher is the value output by cosine similarity, more similar is the premise-hypothesis pair.

By taking a look at the outputs, it seems that cosine similarity works as expected. Fig.3.7 shows contradictory sentence pairs which present low values of cosine similarity, while Fig.3.8 presents the exact opposite situation for sentences classified as entailment.



	premise	hypothesis	label	cosine_similarity
1	these are issues that we wrestle with in pract...	practice groups are not permitted to work on t...	2	0.408248
3	from cockpit country to st ann s bay	from st ann s bay to cockpit country	2	1.000000
8	the streets are crammed with vendors selling s...	vendors have lined the streets with torches an...	2	0.382360
9	north of mytilini stop at the village of moria...	there is nothing special to see in the village...	2	0.263181
13	savonarola burned in florence	florence became savonarola s new home	2	0.408248

Figure 3.7: Cosine similarity for contradiction



	premise	hypothesis	label	cosine_similarity
0	and these comments were considered in formulat...	the rules developed in the interim were put to...	0	0.639010
2	you know they can t really defend themselves l...	they can t defend themselves because of their ...	0	0.500000
6	my own little corner of the world policy wonki...	an example is policy wonking	0	0.645497
10	increased saving by current generations would ...	current generations increased saving would exp...	0	0.923077
14	it will be colossal	it will be gigantic	0	0.750000

Figure 3.8: Cosine similarity for entailment

3.2.3 Polarity

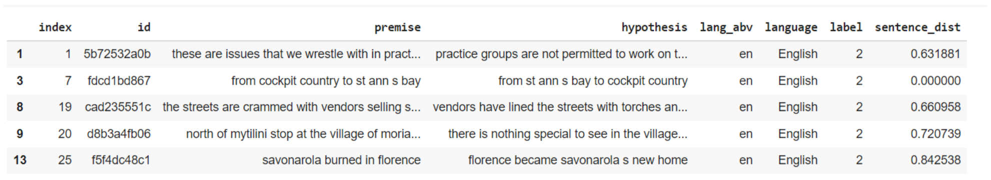
Another feature that has been performed is polarity, which defines the semantic orientation of the sentences, in such a way to determine when the premise and hypothesis sentences are positive or negative. In the context of contradiction analysis, it can be relevant to understand when one sentence is characterized by a positive sentiment and the other one by a negative one, meaning that they can be contradictory. Furthermore, it can be also interesting to define the difference in polarity of the pair of sentences. Let’s think about a situation in which two phrases contain a word which has been identified as being an antonym, such as ”I love big parties” and ”I love small parties”, where ”big” and ”small” are antonyms. Being characterized by the presence of an antonym and by a reasonably high similarity in polarity, the two sentences can be defined more strongly as contradictory.

There are many packages available in Python which enable to use various methods to do sentiment analysis. In this project, polarity has been performed trying two different Python libraries, TextBlob and pysentiment. The approach taken with pysentiment analysis has performed better than Textblob and this is why measures have been performed using pysentiment.

3.2.4 Sentence simililarity

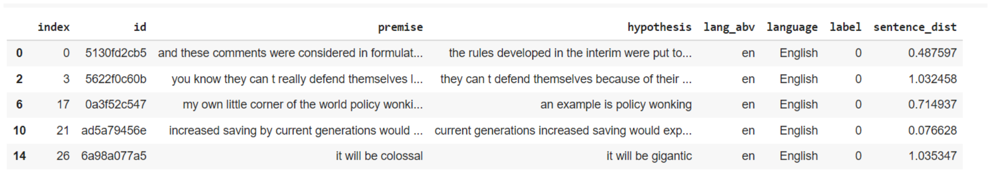
Another performed feature refers to sentence similarity. Word2vec has been used to produce distributed representations of words in sentences using a pretrained model by Google, known as GoogleNews-vectors-negative300.bin, whose vocabulary size is around 3 millions words. The similarity between premise and hypothesis sentences is calculated through Word Mover's Distance.

As it is possible to see in Fig.3.9 and Fig.3.10, distance is bigger when premise and hypothesis are contradicting, while it is smaller when they are not.



	index	id	premise	hypothesis	lang_abv	language	label	sentence_dist
1	1	5b72532a0b	these are issues that we wrestle with in pract...	practice groups are not permitted to work on t...	en	English	2	0.631881
3	7	fdcd1bd867	from cockpit country to st ann s bay	from st ann s bay to cockpit country	en	English	2	0.000000
8	19	cad235551c	the streets are crammed with vendors selling s...	vendors have lined the streets with torches an...	en	English	2	0.660958
9	20	d8b3a4fb06	north of mytilini stop at the village of moria...	there is nothing special to see in the village...	en	English	2	0.720739
13	25	f5f4dc48c1	savonarola burned in florence	florence became savonarola s new home	en	English	2	0.842538

Figure 3.9: Word Mover's Distance Conradictions



	index	id	premise	hypothesis	lang_abv	language	label	sentence_dist
0	0	5130fd2cb5	and these comments were considered in formulat...	the rules developed in the interim were put to...	en	English	0	0.487597
2	3	5622f0c60b	you know they can t really defend themselves l...	they can t defend themselves because of their ...	en	English	0	1.032458
6	17	0a3f52c547	my own little corner of the world policy wonki...	an example is policy wonking	en	English	0	0.714937
10	21	ad5a79456e	increased saving by current generations would ...	current generations increased saving would exp...	en	English	0	0.076628
14	26	6a98a077a5	it will be colossal	it will be gigantic	en	English	0	1.035347

Figure 3.10: Word Mover's Distance Entailment

3.2.5 Negation

At first, negation feature has been implemented by considering a collection of negations, represented by words such as ”not” or ”nothing”, which has been used to count their frequency in premise and hypothesis sentences. Also negation modal verbs haven been added to the collection, such as ”cannot”. It has been taken into consideration that if premise contains a negation and hypothesis does not (and vice versa), they can be defined as creating a contradiction. If measure is 1, it is meaning that it is possible that the two sentences contradict themselves because of negation presence, otherwise they do not. Negation is fundamental in contradiction analysis since a sentence containing a negation word with respect to another sentence that lacks it could reveal a contradiction. However this feature has been then determined using spaCy dependency parsing along with antonyms detection in such a way to perform a better analysis as described in Section. 3.2.6

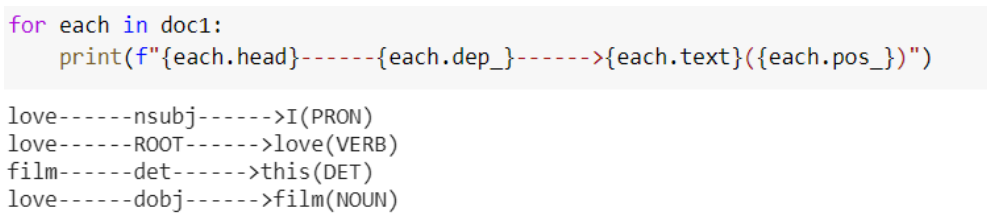
3.2.6 Antonyms

Negation alone is not sufficient to identify contradiction, because one can have negation in a sentence which potentially contradicts another one, but this negation could be accompanied by an antonym meaning that negation is annulled. Let's consider two sentences such as "I love this film" and "I don't hate this film". Their meaning is the same but by considering just negation based on the presence of "not" could be diverted, while if considering also the presence of the antonym "hate" with respect to "love" could define effectively the absence of contradiction in this case. So, determining a feature combining negation and antonyms is fundamental for contradiction analysis. Antonyms have been defined through WordNet and spaCy has been used to identify negation though the existing negation modifier "neg" in the dependency tag schema and to check that root of the two sentences is a verb with its lemma. This feature takes the value of 0 if none of the words in the pair of sentences have antonyms in the other one, while it takes the value of 1 for the opposite situation.

Let's consider the previously cited example to make it clear how the measure performs, where the premise and hypothesis are:

- sentence 1: "I love this film"
- sentence 2: "I don't hate this film"

In Fig.3.11 and Fig.3.12, it is possible to see how spaCy parses and tags the two sentences. Dependency parsing extraction consists in representing the grammatical structure of the sentence, identifying the dependency relationship between head words, known as root words, and their dependents. Furthermore, spaCy maps all tokens with Part of speech (POS) tags. POS consists in marking up a word in a sentence corresponding to a part of speech, such as nouns, verbs, pronouns etc.



```
for each in doc1:
    print(f"{each.head}------{each.dep_}------>{each.text}({each.pos_})")
```

```
love------nsubj------>I(PRON)
love------ROOT------>love(VERB)
film------det------>this(DET)
love------dobj------>film(NOUN)
```

Figure 3.11: Token, syntactic dependency and POS representation for sentence 1.

```
for each in doc2:
    print(f"{each.head}------{each.dep_}------>{each.text}({each.pos_})")
```

```
hate------nsubj------>I(PRON)
hate------aux------>do(AUX)
hate------neg------>n't(PART)
hate------ROOT------>hate(VERB)
film------det------>this(DET)
hate------dobj------>film(NOUN)
```

Figure 3.12: Token, syntactic dependency and POS representation for sentence 2.

In this case, one can notice that sentence 1 presents a root verb, ”love”, while sentence 2 presents a root verb, ”hate”, accompanied by a neg dependency, ”n't”. A better representation of these dependencies is given in Fig.3.13 and Fig.3.14.

```
spacy.displacy.render(doc1, jupyter=True)
```

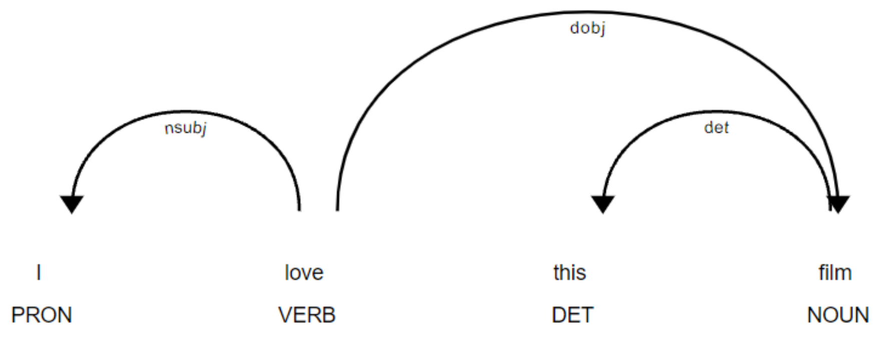


Figure 3.13: Token representation for sentence 1.

```
spacy.displacy.render(doc2, jupyter=True)
```

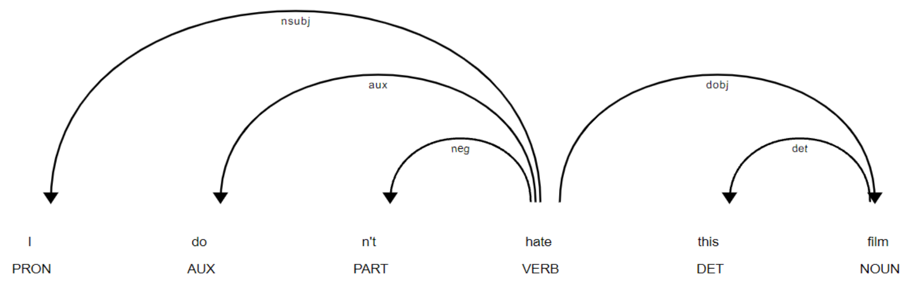


Figure 3.14: Token representation for sentence 2.

The implemented function checks for negation in both sentences and it recognizes that ”love” and ”hate” are antonyms identifying the two sentences as not contradictory and so outputting 0. Furthermore, for performing this measure contractions have not been extended since it has been noticed that spaCy performed better without processing them. In the end, among the 2277 pairs of sentences classified as contradiction, there have been revealed 831 pairs having antonyms feature; while among the ones categorized as entailment 312 of a total of 2427 contain it.

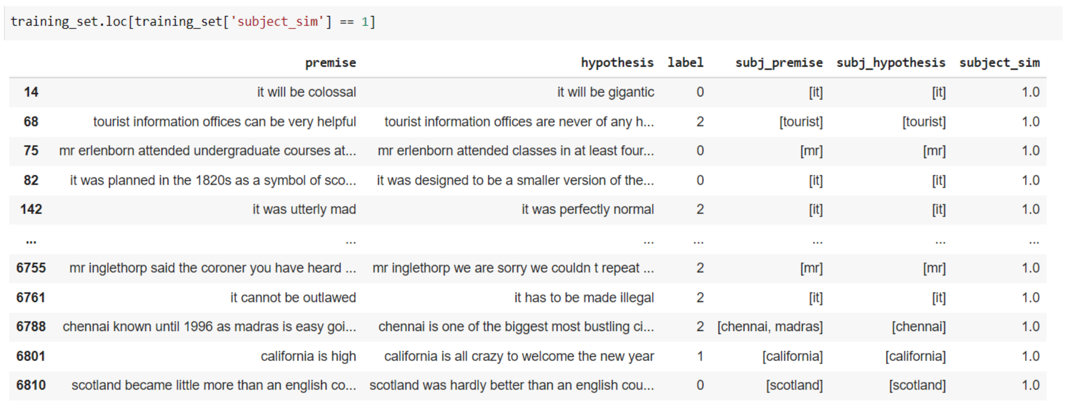
3.2.7 Numeric mismatch

During data analysis, it has been checked the amount of numbers in the dataset. Numeric features consist in searching for pairs of sentences referring to the same entity but with a different number, which could lead to a contradiction. The numeric portion as concerns premise sentences is about 14% in relation to the entire training dataset, numeric features have been performed. If numbers do not exist in both or one of premise and hypothesis sentences, the numeric feature results in 0, otherwise a comparison is made: when numbers for both sentences are equal, the numeric feature is associated with 1; when numbers are different, feature equals -1. Once having performed this feature, it has been noticed that about 95% of the pairs of sentences resulted in not having both numeric features where just 297 pairs of sentences were mapped as having or not having numeric mismatch. This feature has not so be considered in the final datasets for feeding the algorithms for the goal of this reasearch.

3.2.8 Subject-subject similarity

Intuitively, when comparing two sentences in general, especially in the case in which they contradict, it is fundamental that they refer to the same subject. So, it has been performed a feature which identifies the subject of premise and hypothesis sentences. Once it has been checked of being nouns or pronouns, the subject of the premise has been compared with the subject of the hypothesis. Their similarity has been calculated by averaging the similarity between each synset for premise and hypothesis subjects, where a score of 1 represents a perfect match.

By filtering for those subject-subject similitudes resulted being 1.0, one can notice that subject-subject measure performs as expected in achieving the predefined goal, as per Fig.3.15.



```
training_set.loc[training_set['subject_sim'] == 1]
```

	premise	hypothesis	label	subj_premise	subj_hypothesis	subject_sim
14	it will be colossal	it will be gigantic	0	[it]	[it]	1.0
68	tourist information offices can be very helpful	tourist information offices are never of any h...	2	[tourist]	[tourist]	1.0
75	mr erlenborn attended undergraduate courses at...	mr erlenborn attended classes in at least four...	0	[mr]	[mr]	1.0
82	it was planned in the 1820s as a symbol of sco...	it was designed to be a smaller version of the...	0	[it]	[it]	1.0
142	it was utterly mad	it was perfectly normal	2	[it]	[it]	1.0
...	...	...	...	...	...	...
6755	mr inglethorp said the coroner you have heard ...	mr inglethorp we are sorry we couldn t repeat ...	2	[mr]	[mr]	1.0
6761	it cannot be outlawed	it has to be made illegal	2	[it]	[it]	1.0
6788	chennai known until 1996 as madras is easy goi...	chennai is one of the biggest most bustling ci...	2	[chennai, madras]	[chennai]	1.0
6801	california is high	california is all crazy to welcome the new year	1	[california]	[california]	1.0
6810	scotland became little more than an english co...	scotland was hardly better than an english cou...	0	[scotland]	[scotland]	1.0

Figure 3.15: Subject-subject similitude performed on training set

3.2.9 From-to Feature

Furthermore, it has been checked also how many sentences contain the pattern ”from to place”, since the intention behind it would have been searching for sentences matching this pattern and compare the name of places in premise and hypothesis. However, the number of sentences having this format is just about 2% of the entire training dataset. So, this feature has not been considered for this analysis.

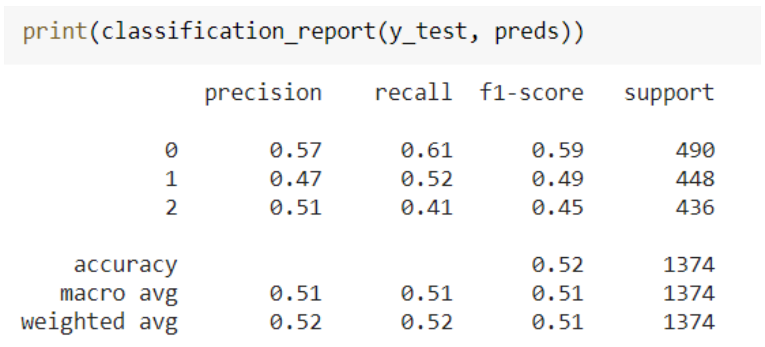
Training and Experimental Results

In this Section, two approaches will be presented, one using the dataset with the previously extracted features and one using word embeddings. Are sentence encoder able to get patterns and lead to contradiction detection?

4.1 Decision Tree Implementation

After having standardized data and divided them into training and testing sets in the ratio of 80:20, Classification Decision Tree has been implemented, defining best tree depth through cross-validation and training it with both Gini and Information Gain for studying the difference in decisions taken by the predictor resulted implementing the two criteria.

As per Fig.4.1, it is possible to look at different performance measures in such a way to measure the success of prediction.



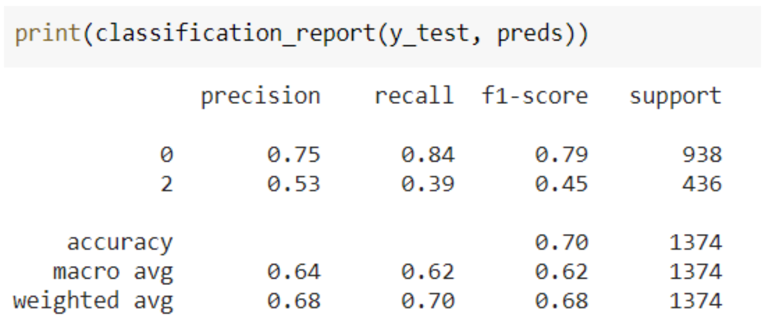
```
print(classification_report(y_test, preds))
```

```
              precision    recall  f1-score   support

           0       0.57      0.61      0.59       490
           1       0.47      0.52      0.49       448
           2       0.51      0.41      0.45       436

    accuracy                           0.52      1374
   macro avg       0.51      0.51      0.51      1374
weighted avg       0.52      0.52      0.51      1374
```

Figure 4.1: Decision Tree Classification Report

As per Fig.4.1, it is possible to see that performance results are not so good in terms of contradiction detection. After multiple tests on all the algorithms, it has been considered that the models were performing better by transforming the problem into a binary problem.

In Fig.4.2,it is possible to see that Precision and Recall performed much better as concerns entailment classification and little better for predicting contradiction in terms of lower FP rate. More in detail, by performing the confusion matrix, it results that there are 171 TP (contradictions predicted as contradictions), 788 TN (neutral/entailment predicted as expected), 150 FP (actually not contradictions predicted as contradictions) and 265 FN (actual contradictions predicted as not being contradictions).



```
print(classification_report(y_test, preds))

              precision    recall  f1-score   support

           0       0.75      0.84      0.79       938
           2       0.53      0.39      0.45       436

    accuracy                           0.70      1374
   macro avg       0.64      0.62      0.62      1374
weighted avg       0.68      0.70      0.68      1374
```

Figure 4.2: Decision Tree Classification Report - Binary

Fig.4.3 represents the Decision Tree, where it is possible to easily visualize and interpret each internal node accompanied by the decision rule that splits the data.

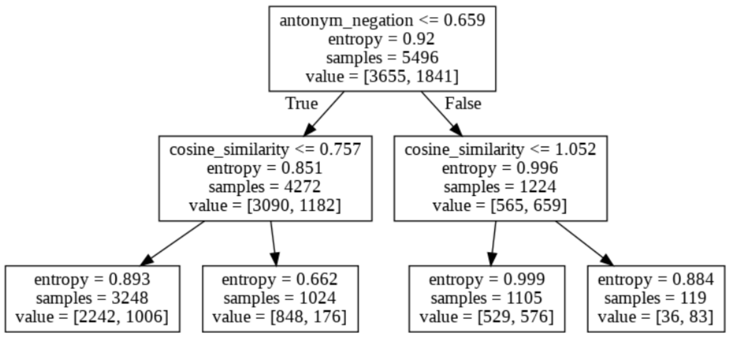


Figure 4.3: Decision Tree Representation

Moreover, in Fig.4.4, it has been created a bar chart representing feature importance scores for the used attributes. These scores allows to better understand the data and the model, where one can observe that antonym_detection and cosine_similarity have been more relevant.

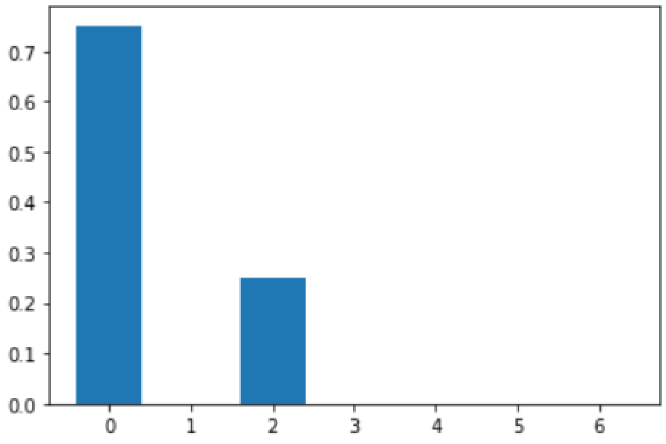
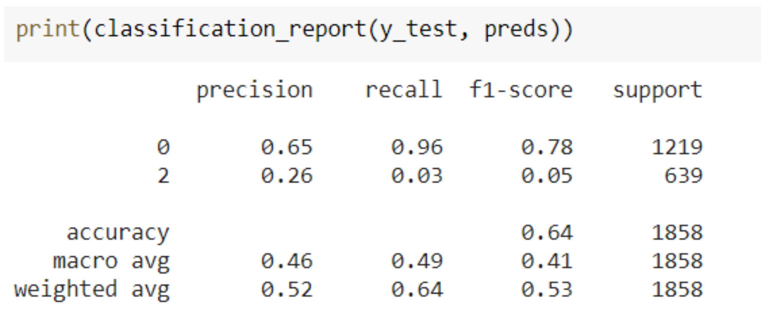


Figure 4.4: Decision Tree Feature Importance

Decision Tree has been performed using word embeddings. Vector representations are learned using Word2Vec, which aims at considering the semantic closeness of the words. As concerns Word2vec, the pre-trained model explained in Section.3.2.4 has been used.

According to Fig.4.5, which defines the results of Decision Tree by Word2Vec encoder, it is possible to notice that Decision Tree performed better in first situation, facing difficulties in predicting contradictions with a high rate of FN.



```
print(classification_report(y_test, preds))

              precision    recall  f1-score   support

           0       0.65      0.96      0.78      1219
           2       0.26      0.03      0.05       639

    accuracy                           0.64      1858
   macro avg       0.46      0.49      0.41      1858
weighted avg       0.52      0.64      0.53      1858
```

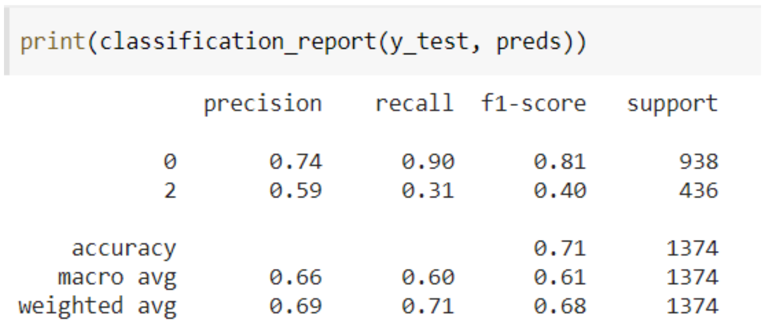
Figure 4.5: Decision Tree Classification Report - Word2Vec

4.2 Random Forest Implementation

As previously defined, Random Forest consists of a grouping of Decision Trees. But how many Trees should be considered? Which size should have each Tree? How many features should be provided for each Tree? To decide how to implement all these hyper-parameters, 10-fold cross validation has been performed in all approaches.

At first, Random Forest has been trained on the dataset with the previously extracted features.

According to the resulted confusion matrix, there have been 844 TN, 94 FP, 302 FN and 134 TP. According to Fig.4.8, it seems that RF performance is similar to DT one, where it is still not good at predicting a high rate of contradictions in the actual class.



```
print(classification_report(y_test, preds))
```

```
              precision    recall  f1-score   support

           0       0.74      0.90      0.81       938
           2       0.59      0.31      0.40       436

    accuracy                           0.71      1374
   macro avg       0.66      0.60      0.61      1374
weighted avg       0.69      0.71      0.68      1374
```

Figure 4.6: Random Forest Classification Report - Dataset with extracted features

By looking at feature importance, the first four features are consistently contributing to predictions:

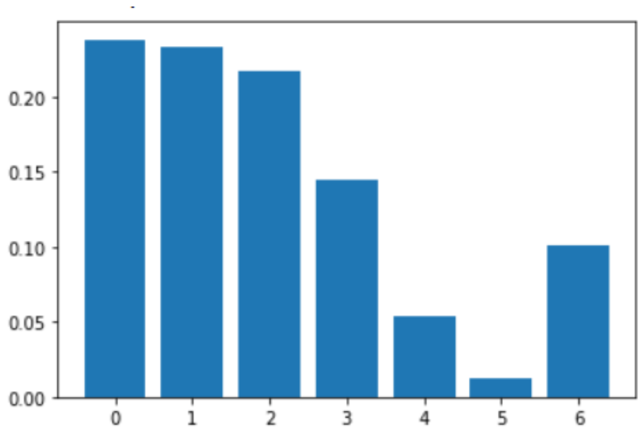
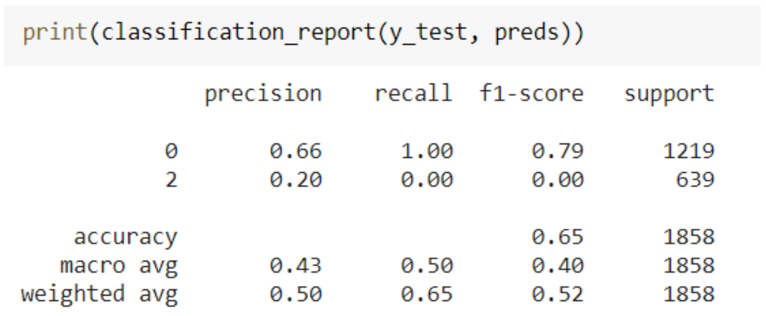


Figure 4.7: Random Forest Feature Importance

The second approach made using word embeddings shows once again that training applied to sentences' vector representations performs really bad since even if achieving an accuracy of 66% it predicts as expected just entailment observations. As one can observe in Fig.4.8, recall is 0 for contradiction, meaning that there is a really high of FP. In fact, by looking at the confusion matrix, just one case has been predicted as expected.



```
print(classification_report(y_test, preds))

              precision    recall  f1-score   support

           0       0.66      1.00      0.79      1219
           2       0.20      0.00      0.00       639

    accuracy                           0.65      1858
   macro avg       0.43      0.50      0.40      1858
weighted avg       0.50      0.65      0.52      1858
```

Figure 4.8: Random Forest Classification Report - Dataset Word2Vec

4.3 ANN Implementation

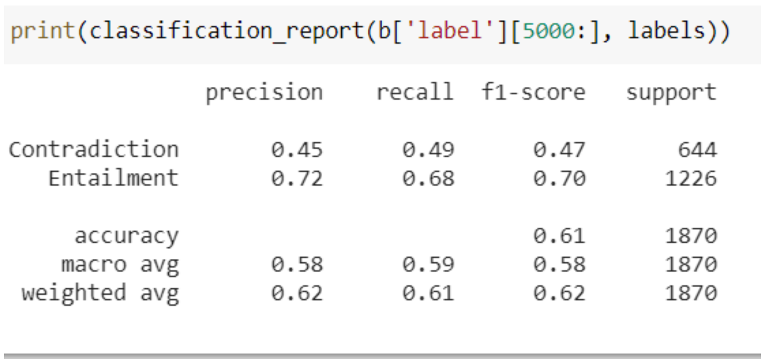
At first Neural Networks have been trained on the dataset with the extracted features, which have been firstly standardized. It has been performed neural neurwork using Keras, using stratified 10-fold cross validation to evaluate the model performance. The model has a single fully connected hidden layer. The activation function used is the

Rectifier activation. As concerns the output layer, it contains a single neuron and it makes use of the Sigmoid activation function in such a way to perform probability output that can be converted to class values. Since the problem has been transformed to binary, the loss function that has been used is the binary crossentropy. CV score present a 68.75% accuracy. Even if accuracy is close to 70%, precision and recall are equal to 0, meaning that the model is not performing well at all.

By implementing Neural Networks after Word2Vec embedding, accuracy results being lower, more precisely 65% with bad results as concerns the other measures. Even if by trying to reduce neurons in such a way to lead the network to pick out the most relevant structure in the input data, the situation does not change so much.

4.4 Sparse Tensor Implementation

Even if Sparse Tensor Classifier is a supervised classification algorithm used for categorical data, it has been tried to instruct the classifier to predict on the attributes previously derived to identify contradictions. It has been really interesting to notice its performance. It can be seen that precision is 45% for contradiction and 72% for entailment, while recall is 49% for contradiction and 68% for entailment as shown in Fig.4.9, presenting so a lower FN and FP rate than the other evaluated classifiers. It has been consequently tried to perform it also by considering the original dataset labelled with entailment, neutral and contradiction, but performance measures were lower.



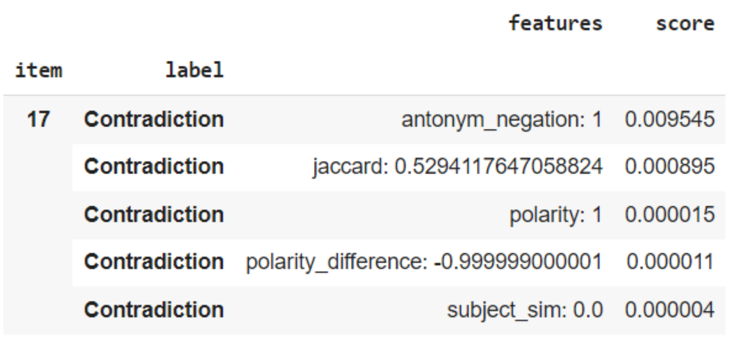
```
print(classification_report(b['label'][5000:], labels))
```

```
               precision    recall  f1-score   support

Contradiction       0.45      0.49      0.47       644
   Entailment       0.72      0.68      0.70      1226

     accuracy                           0.61      1870
    macro avg       0.58      0.59      0.58      1870
 weighted avg       0.62      0.61      0.62      1870
```

Figure 4.9: STC Classification Report

STC returns a tuple comprising the predicted labels, probability for each class and also explainability for each feature, in such a way to go deeper into the investigation of the model performance.

In Fig.4.10, by taking an example of TP, which so has been predicted as contradiction, where the probability of Contradiction is 0.64, and so higher with respect to the entailment class, it is possible to notice that antonym negation, Jaccard and polarity contributed most to the result. This particular situation highlights one of the thoughts of the research, in which by having an antonym in a sentence but not in the other one and by having a high polarity could reveal a contradiction.



item	label	features	score
17	Contradiction	antonym_negation: 1	0.009545
	Contradiction	jaccard: 0.5294117647058824	0.000895
	Contradiction	polarity: 1	0.000015
	Contradiction	polarity_difference: -0.999999000001	0.000011
	Contradiction	subject_sim: 0.0	0.000004

Figure 4.10: Explainability Contradiction

Since STC supports also JSON input data, it has been created a list of dictionaries, containing hypothesis, premise and label. Premise and hypothesis are lists of tokens of the words of the sentences. This kind of implementation has been revealed of being low performant, especially as concerns entailment classification.

Concluding Remarks

The aim for this project was to reveal contradictions in text. To achieve the goal, it has been tried to find patterns in sentences at lexical, syntax, semantic level. Potential relevant features for contradiction such as Jaccard similarity, cosine similarity, polarity, antonyms have been performed. Best predictive performances have been achieved by Decision Tree and Sparse Tensor Classifiers, where a further analysis can be carried out to reduce more the number of both FP and FN. It has been also performed another approach, where classifiers have been fed using the representation of data based on word embedding, more precisely through Word2vec. It has been seen that this way of representing relations of words does not catch the subtleties of the language and cannot help the predictor to understand and consequently output contradictions.

Bibliography

[1] Kaggle.com,*Contradictory, My Dear Watson, Detecting contradiction and entailment in multilingual text using TPUs.* Available at: https://www.kaggle.com/c/contradictory-my-dear-watson/data

[2] D. Giampiccolo et al., *The Fourth PASCAL Recognizing Textual Entailment Challenge*, 2009.

[3] E.Guidotti, A.Ferrara *Categorical Data Insipired to Quantum Physics*, 2021.