2. Kaggle Text to Emotion Dataset

Andrea Cantore Student ID: 2308193, **Davide Moricoli** Student ID: 2308158, GitHub: https://github.com/Cavolino/TweetsToEmotions

***Index Terms*—Emotion categorization and classification, Natural Language Processing, Text Mining, Empath Categorization, LDA Topic Modeling, BerTopic, FuzzyWuzzy String Matching, Semantic Analysis, Antonym Relation, BERT Embeddings, Machine Learning, Random Forset, XGBoost.**

***Abstract*—In this project, we analyzed a collection of 40K Twitter records classified in 13 different categories. Our investigation spans different methodologies to enhance the understanding of the emotions in text data.**

We conducted an initial exploration of the dataset. Then we categorized the different tweets using different approaches: Empath, LDA, and BerTopic. Given as input the set of tweets belonging to the same category, the common goal of these three approaches was to identify keywords and check their compatibility with the emotion corresponding to the input category.

We then investigated in more detail the tweets of each emotion. In particular, we evaluate the redundancy and the diversity between tweets of the same category. We also studied the agreement of each record with the antonyms of the emotion associated with it.

Finally, we implemented several algorithms to increase the classification rate of the different categories.

I. Introduction

Emotion detection from text is one of the challenging problems in Natural Language Processing. The reason for this is the lack of labeled datasets and the multi-class character of the problem. Humans experience a wide range of emotions, and it is difficult to collect enough data for each feeling, resulting in the problem of class imbalance.

Emotion analysis in the context of social networks is a particularly difficult task. The intricacy of this task increases in the context of platforms such as Twitter, where users were limited to a character restriction of 140 characters per message. As a result, a significant number of Tweets are created using abbreviations and a specialized language that is difficult to decipher and read precisely.

In the context of this particular study, the dataset has been labeled across 13 distinct categories. The primary objective of this research is to delve deeply into this pre-existing categorization scheme. The aim is to comprehensively analyze and gain insights into the underlying principles governing these categories. Furthermore, the overarching goal of this project is to enhance the efficacy of classification methodologies through the application of machine-learning techniques.

A. Data Exploration

The dataset used in this paper was downloaded from Kaggle. It contains 40K Twitter records classified into 13 categories/emotions: *neutral*, *worry*, *happiness*, *sadness*, *love*, *surprise*, *fun*, *relief*, *hate*, *empty*, *enthusiasm*, *boredom*, and *anger*. The dataset dates back to 2016 and was provided by Appen [1].

1) Distribution: The first step was an initial exploration of the dataset. Table 1 shows the main information about the dataset and highlights the absence of null rows.

#	Column	Non-Null	Count	Dtype
0	tweet_id	40.000	non-null	int64
1	sentiment	40.000	non-null	object
2	content	40.000	non-null	object

TABLE I: General dataset info.

2) Length: In 2016, Twitter had a character limit of 140 characters for tweets. This restriction was a key component of the platform and affected the way users created and distributed their messages. This character restriction promoted concision and brevity, and it gave rise to innovative ways of expressing oneself, about which this paper will go into more detail in later sections. In Figure 1 is possible to identify a spike at 140 characters in length. Many users adapted to this limit and became skilled at crafting concise messages to convey their thoughts effectively.

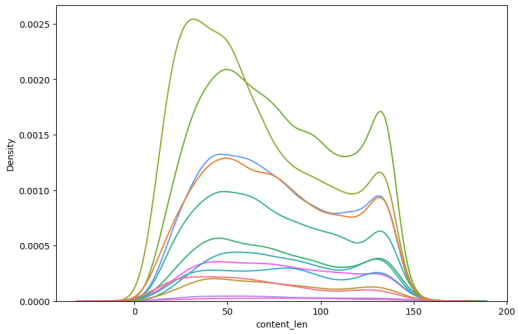

Fig. 1: Distribution of tweet length by emotion class.

II. Methodology

A. Pre-process

As mentioned before, due to the limitation of the length of the tweets and the nature of the language generally used on social media, the pre-processing carried out on the dataset was treated in depth. Initially, all contractions were expanded, and then all words preceded by '@' (which usually concern tags of other people on the platform) were removed. Naturally, all links, some special characters, and stopwords in the English dictionary were removed. After this, 'TweetTokenizer' was used for tokenization.

TweetTokenizer is a specialized tokenizer used in natural language processing (NLP) and text analysis, for handling text data from social media platforms, especially Twitter. It is a part of the NLTK library in Python. Finally, punctuation has been removed. Figure 2 shows the wordcloud computed for each category in the dataset.

Fig. 2: Wordcloud of each category.

B. Data preparation

To complete the different tasks, an additional dataset was created that concatenates all pre-processed tweets of the same category. This way, in the new file, each emotion is associated with all the different tweets labeled with that category in the initial dataset.

C. Empath

Empath is a Python tool that analyzes text across lexical categories as well as generates new lexical categories for analysis.[2] Empath generates new categories by providing a few keywords, following which it obtains a greater number of words with topics similar to the keywords. It has been used to output the key categories of each of the 13 emotions. Empath's output is a dictionary that associates the categories and the relevance score to the input. All the tweets related to a category were passed as input.

D. LDA

Latent Dirichlet Allocation (LDA) is employed to discover hidden topics within a collection of documents. It assumes that documents are mixtures of topics and that words in documents are attributable to those topics. The *sklearn.decomposition.LatentDirichletAllocation* module in scikit-learn provides an implementation of this algorithm in Python, allowing the application of LDA to the text data. LDA has been used to find an array of three keywords associated with each initially labeled emotion. The goal is to find the correspondence between the category and this array, and then analyze the similarity between those words.

E. BerTopic

BerTopic is a Python library that offers a powerful and efficient way to perform topic modeling on text data using state-of-the-art natural language processing techniques.[3] With BerTopic, it is possible to discover meaningful topics within text documents, making it a valuable tool for text analysis, document clustering, and content organization. In this case, BerTopic was used to find different topics associated with each category. Those topics were then ranked with the following formulas:

$$rank[c][t] = \frac{P(t,c)}{\#Tweets\ of\ C} \tag{1}$$

$$\text{Where: } P(t,c) = \sum_{i}^{\#TopicInC} (p_i|t,c) \tag{2}$$

Formula (1) represents the percentage of a topic t within a category c. In this way, it is possible to rank each topic and find the one most present and most likely within a given emotion. #TweetsOfC is the number of Tweets with the category c labeled.

Formula (2) calculate the total percentage of a topic t belonging to a category c. BerTopic in fact returns a topic with attached probability for each tweet associated with a category. #TopicInC is the number of the topic t in the category c associated.

F. FuzzyWuzzy

FuzzyWuzzy is a Python library that provides simple and effective tools for string matching and comparison. It uses various algorithms to calculate the similarity between strings. With FuzzyWuzzy, it is possible to compare strings that are not exactly the same, allowing for flexibility in handling textual data that may contain errors, typos, or inconsistencies. FuzzyWuzzy was used to calculate the ratio score of every pair of records belonging to the same category.

G. Word Sense Variation

WordNet is an English lexical database that is used to identify conceptual connections between words. It is an NLTK corpus reader, utilized to obtain the number of senses for each word, to find semantic relations between a term belonging

to a set that is defined by another term and the latter, and also to find words' antonyms. Words that are similar to one another are semantically disambiguated because WordNet groups words according to their meanings.

H. Wu&Palmer Semantic Similarity

Wu and Palmer's semantic analysis is a method used in NLP to measure the similarity between two words or concepts based on their semantic relationships. It is a similarity measure that takes into account the depth of their common ancestors in a hierarchical structure. If two words share a relatively high common ancestor in the hierarchy, they are likely to be more closely related in meaning. The similarity score produced by Wu and Palmer's method is a value between 0 and 1, with 1 indicating maximum similarity and 0 indicating no similarity.

Using this method, an evaluation of the proximity between the different categories will be addressed. To do this, the antonyms of each category were used. Specifically, the following formula was used to assess the similarity between categories C_i and C_j:

$$Sim(C_i, C_j) = 1 - \frac{1}{2}\Bigg[\max\big(Sim_{wp}(C_i, S_{i1}), \ldots, Sim_{wp}(C_i, S_{in})\big) + \max\big(Sim_{wp}(C_j, S_{j1}), \ldots, Sim_{wp}(C_j, S_{jn})\big)\Bigg] \quad (3)$$

where S_{1i}, S_{2i}, ..., S_{n_i} is the list of antonyms for the category C_i. S_{1j}, S_{2j}, ..., S_{n_j} is the list of antonyms for the category C_j.

As shown by the formula (3), for each category the list of the antonyms is identified and then Wu and Palmer similarity is used to evaluate the similarity between the category and every antonym.

I. BERT

BERT, which stands for "Bidirectional Encoder Representations from Transformers," is an NLP model developed by Google and it is based on the Transformer architecture, a deep learning model. BERT is pre-trained on a large corpus of text (such as Wikipedia and the BookCorpus) by considering both the left and right context for each word in a sentence. In this way, it is able to capture richer contextual information and a deeper understanding of language semantics. The original BERT model is known as "BERT-Base," and it has 110 million parameters. BERT base model (uncased) has been utilized to test the extent to which the records of each category agree with the antonym keywords. The model used is "uncased", so it does not make a difference, for example, between english and English. In particular, the model has been used to compute the embedding vector of each record R and the embedding vector of the antonyms. Formula (4) has been utilized to compute the Similarity between a category and its records.

$$Sim(C_i, R) = 1 - \max_{j=1,n}(Cos(BERT(R), BERT(S_{ij}))) \quad (4)$$

S_{i1}, S_{i2}, ..,S_{in} are the antonyms of the category C_i. The cosine similarity between the two embedding vectors is the normalized measure.

J. Random Forest classifier & XGBoost

Random Forest is a machine learning algorithm that is commonly used for classification tasks, including text classification in NLP. It is an ensemble learning technique that combines multiple decision trees to make more accurate predictions. Before applying the Random Forest classifier, the text has been preprocessed. This involves tokenization, removing stop words, and handling special characters, hashtags, and mentions. Then the text has been vectorized, i.e. convert the processed text data into numerical features. Techniques utilized in this step have been Bog of Words (BoW) and TF-IDF. The performance of a Random Forest classifier was initially investigated using 500 TF-IDF size as a feature vector. The criterion for splitting nodes has been set as 'gini' and the number of estimators has been set to 150. Moreover, has been explored different choices for feature vector sizes. Has been used a training-testing split of 80-20%, with the addition of 10-fold cross-validation. This cross-validation method will provide a robust assessment of the classifier's performance and help avoid overfitting. Following the initial Random Forest classifier has been further investigated the potential of an XGBoost classifier. XGBoost is a gradient-boosting algorithm that is known for its high performance, scalability, and ability to handle complex, high-dimensional data. The number of estimators was set to 150, the maximum depth to 5, and the learning rate to 0.1. Also in this case has been used a training-testing split of 80-20%. To evaluate the performances of the classifiers has been used standard evaluation metrics like accuracy and confusion matrices. Also, the ROC curve has been plotted for the XGBoost.

K. Feature Vectors

TF-IDF (Term Frequency-Inverse Document Frequency) and Bag of Words (BoW) are two common techniques used for feature extraction in NLP and machine learning tasks. They both aim to convert text documents into numerical feature vectors. Bow represents text documents as vectors, where each element of the vector corresponds to a unique word in the entire corpus of documents. The values in the vector indicate the frequency of each word in the document. BoW ignores the order and structure of words in the text and treats each word independently. TF-IDF is a more sophisticated feature extraction technique that not only counts the frequency of words but also assigns a weight to each word based on its importance in the document and its uniqueness across the entire corpus. TF-IDF calculates the Term Frequency (TF), which is the frequency of a word in a document, and the Inverse Document Frequency (IDF), which measures how unique or rare a word is across the entire corpus. The TF-IDF score for a term in a document is the product of its TF and IDF values. TF-IDF gives higher scores to terms that are important within a document but not too common across the

entire corpus. It is commonly used for information retrieval, text mining, and search engines, as well as text classification tasks.

$$TF(t,d) = \frac{(NT)}{(TD)} \tag{5}$$

Where:

- ”NT” represents the number of times term t appears in document d.
- ”TD” represents the total number of terms in document d.
- “t” represents the word.
- “d” represents the document.

$$TF(t,d) = log(\frac{N}{(ND)} \tag{6}$$

Where:

- ”ND” represents the number of documents in which the term t appears.
- ”N” represents the total number of documents in the corpus.

$$TF - IDF(t,d) = TF(t,d) \cdot IDF(t) \tag{7}$$

The result of the TF-IDF calculation is a feature vector in which each element corresponds to a term and the value represents the TF-IDF weight of that term in the document. This vector can then be used for textual analysis purposes such as classification.

III. Results and Discussions

A. Categorization

This section will discuss the results and the differencies between three approaches: Empath, LDA and BerTopic. Using these methods in conjunction can provide a comprehensive and complex understanding of emotions, allowing for a more robust analysis and interpretation of the dataset.

1) Empath: In Table (II), it is possible to see all the keywords found by Empath and Figure 3 shows the relevance score of the keywords for each emotion.

In blue the representation of the relevance point of the keyword. In red the average value of all the relevant points.

The graphs obtained appear to have an exponential-like shape. It follows that the mean in these values will be significantly lower than the peak value, but high enough to exceed the smaller values. For this reason, it has been selected keywords with scores above the average as relevant, so as to remove all those unimportant keywords with respect to the related category. These data and new categories were studied to understand their relevance and similarity to the prelabeled

Sentiment Category	Key Categories
empty	office, dance, money, wedding, domestic_work, sleep, cold, and more
sadness	money, wedding, domestic_work, sleep, cold, hate, family, and more
enthusiasm	help, money, wedding, domestic_work, sleep, cold, family, and more
neutral	help, office, dance, money, wedding, domestic_work, and more
worry	money, wedding, domestic_work, sleep, medical_emergency, and more
surprise	dance, money, wedding, domestic_work, sleep, cold, happiness, and more
love	dance, wedding, domestic_work, sleep, cheerfulness, family, and more
fun	dance, wedding, domestic_work, sleep, cold, family, vacation, and more
hate	office, money, domestic_work, sleep, cold, hate, occurrences, and more
happiness	dance, wedding, domestic_work, sleep, cold, cheerfulness, and more
boredom	money, domestic_work, sleep, cold, hate, family, vacation, and more
relief	wedding, domestic_work, sleep, cold, cheerfulness, and more
anger	office, money, domestic_work, sleep, cold, hate, family, and more

TABLE II: Sentiment Categories and Key Categories

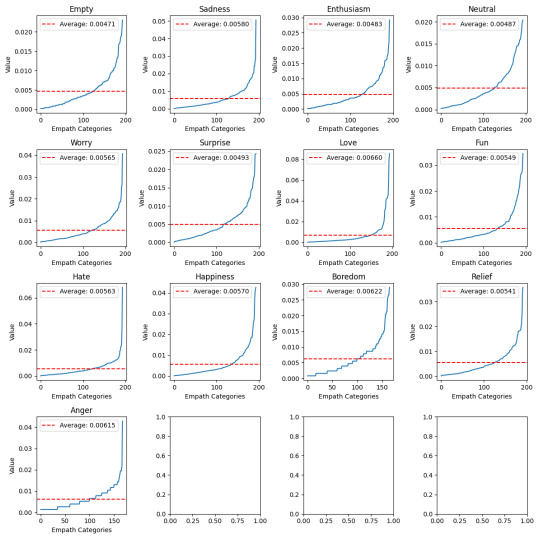

Fig. 3: Descrizione dell’immagine

emotions. As a first step, string matching was studied. As can be seen from Table (III) a flag was placed there to recognize if within the Empath categories results in a perfect match with the related emotion. Of the 13 total categories only 5 have a perfect match, thus a 38.46%. It is not a very good result but this type of test is really stringent.

More worrisome, however, turns out to be the result provided by a semantic analysis.

Semantic analysis was done by comparing the synsets of each category with the synsets of each keyword. In the comparison, it was specifically searched whether there is a hypernyms or hyponyms dependency between the compared

Emotions	Flag	Hyponyms	Hypernyms
empty	False	-	-
sadness	True	-	-
enthusiasm	False	-	-
neutral	False	-	-
worry	False	-	fear, business, eating
surprise	True	-	-
love	True	love	love
fun	True	-	-
hate	True	-	-
happiness	False	-	cheerfulness
boredom	False	-	-
relief	False	-	-
anger	False	-	-

TABLE III: Empath Sentiment Categories

words. Again in Table (III) can be seen under the hypernyms and hyponyms columns the various. Only for *worry*, *love* and *happiness* were semantic matches found. This was not an expected result. While perfect matching is more restrictive, the semantic test is more flexible and should report more results.

2) LDA: In this case, for each category, three keywords were searched using LDA. This allows to analyze, as before, the matching by studying the mapping in two different levels: string matching and semantic analysis. In Table (IV) the result of these analyses can be seen.

Emotions	LDA Keywords	Flag	Hyponyms	Hypernyms
empty	go, get, day	False	-	-
sadness	miss, sad, day	False	-	-
enthusiasm	good, go, want	False	-	-
neutral	get, going, day	False	-	-
worry	day, going, get	False	-	-
surprise	oh, get, day	False	-	-
love	happy, day, love	True	love	love
fun	going, fun, lol	True	-	-
hate	like, suck, hate	True	-	-
happiness	happy, good, day	False	-	-
boredom	go, work, bored	False	-	-
relief	thanks, good, day	False	-	-
anger	going, know, get	False	-	-

TABLE IV: LDA Sentiment Categories

As a result, it is obtained that only three categories appear flagged in the LDA keywords, *love*, *fun* and *hate*. Moreover, in the semantic analysis, only *love* has a hypernyms and hyponyms relationship with another word within the 3 keywords.

These results are partially satisfactory. Three words as keywords are few to describe an entire category but a 23.08% perfect match was obtained. Regarding hypernyms and hyponyms, on the other hand, a low rate was obtained. The only word that was found was *love* with apparently itself, the same as in the Empath analysis. Actually the meaning of *love.v.01* is hypernym of *love.v.03*, which is why it can be seen in the Table (IV). This result seems to go along with Empath's result. Fair success in perfect matching but a poor result in semantic analysis.

3) BerTopic: To delve even deeper into categorization, BerTopic was used. For each category, the BERTopic model was fitted through the tweets belonging to the category. In this way, topics and associated probabilities were found.

With this data, the percentage that a random tweet taken within a given category would belong to a topic was then calculated. Formula (1), in fact, allows to estimate the percentage of a tweet belonging to topic t within category c. In this way, it is possible to rank each topic and find the one most present and most likely within a given emotion.

In Table (V) it is possible to see the ranking of the topics in the category *neutral*. The ranking is represented as a Python dictionary with topic number key and topic probability value.

Topic Number	Percentage
0	3.54
1	1.66
3	1.58
4	1.42
2	1.35
6	1.14
7	0.95
5	0.80
8	0.73
12	0.68
...	...

TABLE V: BerTopic, *neutral* Topics

In Figure 4, on the other hand, can be seen the visualization provided by BerTopic of the previously fitted model. It can be seen that several clusters of topics are present and that the size is larger at the correspondence of a higher percentage. The circle in red in the first graph represents topic 0 in *neutral* category. Can be noticed that in Table (V), topic 0 is the one with the highest percentage ranking calculated previously.

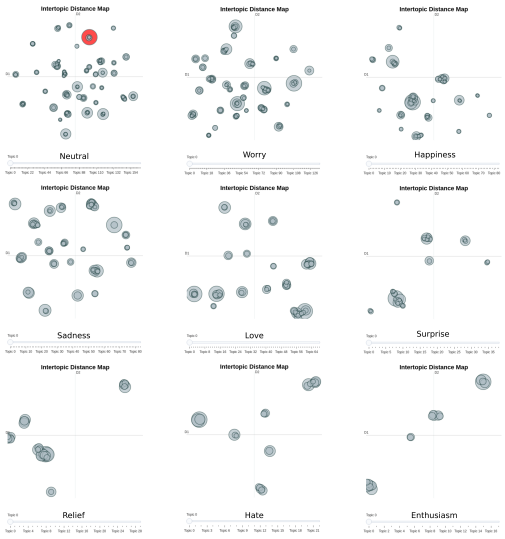

Fig. 4: Bert Cluster Graphs.

It should be noted that not all categories are displayed in Figure 4. This is because the BerTopic function *visualize_topics()* is used to display these graphs. However, when there are too few topics, calling this method results in a "Too less arguments" type error.

Finally, string matching between categories and words within each topic is done. In fact, Table (VI) presents a dictionary where the key represents the category and the associated value is a tuple array containing the topic number and ranking of matching between the category and a word within the topic.

Emotions	**[Topic Number, Percentage]**
empty	[]
sadness	[9, 1.11]
enthusiasm	[]
neutral	[]
worry	[74, 0.19], [91, 0.19]
surprise	[]
love	[1, 2.29], [15, 0.88], [0, 0.85]
fun	[0, 98.77]
hate	[1, 3.95], [2, 2.85], [6, 2.63], [9, 2.14]
happiness	[21, 0.64]
boredom	[]
relief	[]
anger	[]

TABLE VI: BerTopic, Emotion in Topics

In this case 6, emotions out of 13 were perfect matching with the words inside the topics, 46,15%. Then, as in the previous studies, a semantic analysis was carried out by comparing hypernyms and hyponyms. In Table (VII), again is summarized the informations such as perfect matching flags, hyponyms and hypernyms.

Emotions	**Flag**	**Hyponyms**	**Hypernyms**
empty	False	-	-
sadness	True	feeling, feeling	depression
enthusiasm	False	-	-
neutral	False	-	-
worry	True	mind	eat, eating, ate
surprise	False	impressed, attacked	-
love	True	love, loved, couple...	love, loved, crush...
fun	True	-	-
hate	True	-	-
happiness	True	feeling	-
boredom	False	-	-
relief	False	help, change	-
anger	False	-	-

TABLE VII: BerTopic Sentiment Categories with Same Category, Hyponyms, and Hypernyms

It can be seen that for perfect string matching the same results are obtained as Empath. On the other hand, as far as semantic analysis is concerned, a slight improvement can be drawn but still a different and worse result than expected.

B. Category Analysis

In this section, will be analyzed the content of tweets associated with a given category. To do so, initially, the redundancy and the diversity present in each category by looking into string matching will be evaluated. Then will be assessed the agreement of the records of each category with the antonym keywords.

Emotions	Min Ratio Score	Max Ratio Score	Average Ratio Score
empty	0	100	20.92
sadness	0	100	23.18
enthusiasm	0	100	23.35
neutral	0	100	21.29
worry	0	100	23.09
surprise	0	100	22.45
love	0	100	23.69
fun	0	100	23.36
hate	0	100	23.13
happiness	0	100	23.18
boredom	0	65	24.43
relief	0	100	23.17
anger	0	52	23.14

TABLE VIII: FuzzyWuzzy Sentiment Category Data

1) Redundancy and Diversity Evaluation: To evaluate the extent of redundancy and diversity present in each category, the fuzzywuzzy string matching package was used to calculate the ratio score of every pair (without repetition) of records belonging to the same category. Table (VIII) shows the output of the fuzzywuzzy score for each category. From this, it can also be seen that a high minimum ratio score may indicate a high redundancy of records within the category. Conversely, a small value could be indicative of high diversity. In fact, from Table (VIII) can be seen that *anger* and *boredom* have a significantly lower max ratio score than the other categories. This means that the two categories mentioned above are characterised by a high diversity within the respective tweets.

To go deeper into this analysis, the distribution of ratio scores for each category was first plotted (Figure 5) and then this distribution was divided into 10 bins.

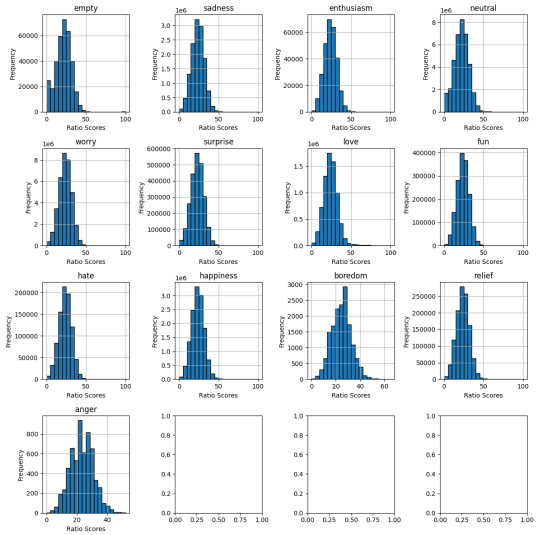

Fig. 5: Distribution of ratio scores for each category.

In Figure 6 is possible to see a barplot that visualizes the distribution of FuzzyWuzzy ratio scores beyond a specified threshold. The x-axis of the plot represents the intervals or bins created to divide the range of ratio scores. Each bar in the plot corresponds to one of these intervals. The intervals are created based on the minimum and maximum values of

Sentiment Category	Min	Max	Mean
neutral	0.20	0.86	0.52
worry	0.05	0.88	0.57
happiness	0.12	0.71	0.50
sadness	0.0	0.85	0.54
love	0.11	0.83	0.54
hate	0.18	0.91	0.55
empty	0.07	0.82	0.48

TABLE IX: Similarity between each category and its records

the ratio scores and are evenly spaced. The y-axis of the plot represents the cumulative percentage of data points that fall into each interval. It shows the relative proportion of data points within each interval with respect to the total number of data points. Each bar in the plot corresponds to one of the intervals. The height of the bar represents the cumulative percentage of data points that fall into that interval.

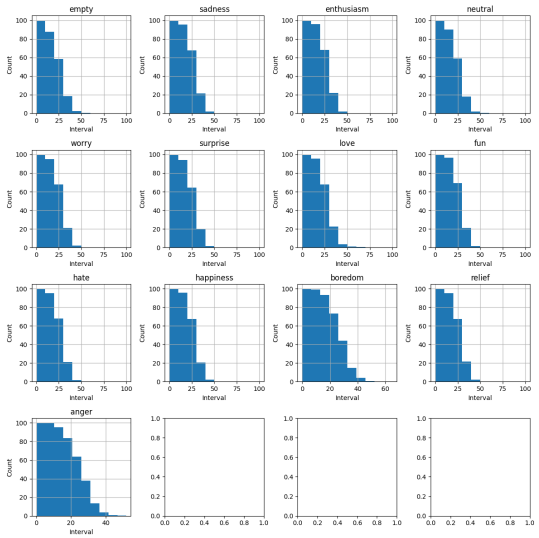

Fig. 6: Distribution of FuzzyWuzzy ratio scores beyond a specified threshold.

The graphs shown in Figure 6 help to understand the absence of redundancy between the tweets in each category. A strongly decreasing graph signifies a strong diversity between the tweets. Although the graphs of anger and boredom seem to decrease more slowly, the range of ratio scores found must be taken into account, as commented above. Consequently, in general, it can be said that the majority of the data within each category is associated with a relatively low fuzzywuzzy. This implies an absence of redundancy for each emotion.

2) Category-Record Agreement: The Table (IX) shows the results for the calculated similarity between each category and the corresponding records (Formula (4)). During the implementation, it was decided to omit categories with no antonyms within Wordnet. The results obtained are linear for each category. The average found represents neither excessive similarity nor excessive diversity.

C. Proximity Evaluation Between Categories

In this section, an evaluation of the proximity between the different categories will be addressed. To do this, the antonyms of each category were used. Specifically, the Formula (3) was used to assess the similarity between categories C_i and C_j:

In Table (A.XIV) the results concerning Formula (3) have been represented. It was chosen to set the value -1 in cases where both categories analyzed do not have associated antonyms. For most categories, a result in line with expectations was obtained (e.g. Sim(”happiness”, ”sadness”) = 0.15 and Sim(”hate”, ”love”) = 0.14). There are some curious results such as the similarities of *worry* being very high. Undoubtedly the list of antonyms present for each category turns out to be a decisive factor in the reported results (especially in the case of similarities between a category without antonyms and one with).

D. Classification Task

As mentioned above, the performance of a Random Forest classifier was initially investigated using 500 TF-IDF size as feature vector. The criterion for splitting nodes has been set as ’gini’ and the number of estimators has been set to 150. The accuracy found on the test set is 31.46% and the configuration matrix is shown in Figure 7.

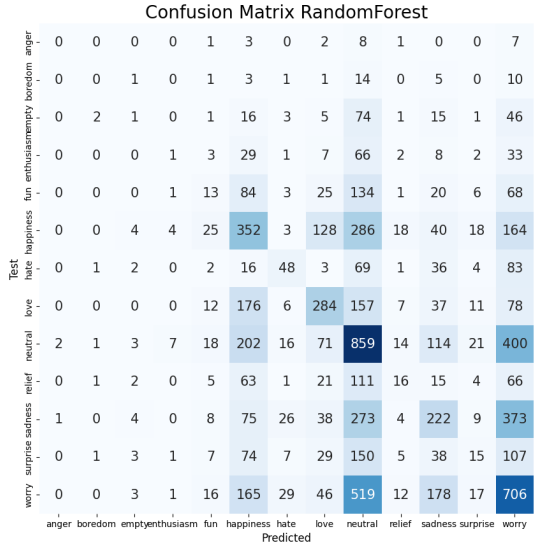

Fig. 7: Confusion Matrix of the Random Forest classifier.

Next, an XGBoost classifier was implemented. The number of estimators was set to 150, the maximum depth to 5, and the learning rate to 0.1. The accuracy achieved on the test set in this case is 33.17%. Figure 8 shows the configuration matrix concerning this last mentioned implementation.

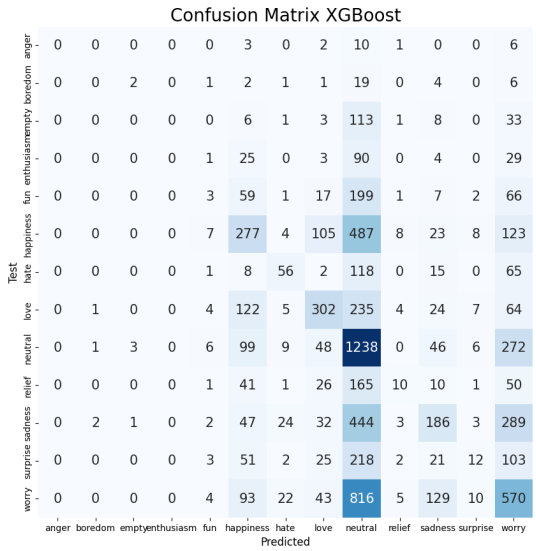

Fig. 8: Confusion Matrix of the XGBoost classifier.

In Figure 9 is shown the ROC curve to illustrate the performance of the XGBoost classifier.

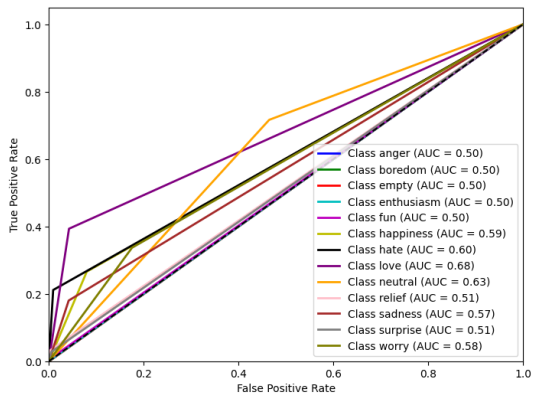

Fig. 9: ROC curve with XGBoost.

As can be seen from the reported values for accuracy, XGBoost seems to perform better. This is in line with the state-of-the-art, as the latter model seems to perform better on high-dimensional and imbalanced class datasets than the Random Forest classifier. Furthermore, from the ROC curve depicted in Figure 9, it is important to highlight that the two categories with a greater area under the curve than the others actually belong to two of the most represented classes within the dataset

Moreover, has been explored different choices for feature vector sizes and alternative feature vectors, aside from TF-IDF. The classification accuracy for each class has been evaluated and reported in terms of the Root Mean Square Error (RMSE) in Table (X). Figure 10 shows a graph for each category where the performance in terms of RMSE of the two types of feature vectors at varying feature vector sizes is represented. As can also be seen from an initial qualitative analysis, it is not possible to identify either an increase in performance as

	TF-IDF			Bag of Word		
	100	300	500	100	300	500
anger	8.852	8.992	9.020	8.723	8.309	8.929
boredom	7.979	7.457	7.886	7.574	7.730	7.230
empty	6.936	7.288	7.343	7.298	7.230	7.434
enthusiasm	5.973	5.924	5.803	5.762	5.647	5.543
fun	4.868	4.755	4.831	5.007	4.938	4.778
happiness	3.629	3.553	3.461	3.635	3.630	3.671
hate	3.336	3.758	3.846	3.539	3.816	3.873
love	2.290	2.177	2.123	2.258	2.278	2.289
neutral	2.143	2.354	2.421	2.277	2.334	2.425
relief	2.550	2.728	2.647	2.532	2.589	2.561
sadness	2.467	2.393	2.370	2.474	2.555	2.623
surprise	3.541	3.586	3.594	3.392	3.440	3.404
worry	3.792	3.564	3.525	3.873	3.774	3.708

TABLE X: RMSE

the vector size increases, or one vectorizing technique better than the other in each category.

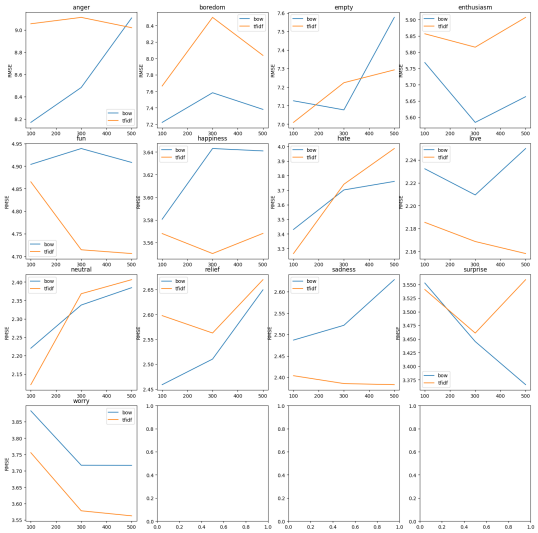

Fig. 10: RMSE Comparison for each Category

In addition, a further analysis was carried out to investigate a possible correlation between the number of tweets in each category and the RMSE calculated for that category by the best combination of feature vector size and type of feature vector. To do this, the RMSE was divided by the number of tweets to obtain a normalized RMSE. The results are shown in Table (XI). There clearly appears to be an increase in performance (lower normalised RMSE) as the data available for the category increases.

1) Reduced Dataset: The already cited imbalance in this dataset poses a significant challenge for sentiment analysis. Imbalanced datasets can lead to biased model training, as the machine learning algorithms may struggle to generalize well to underrepresented categories. To address this issue has been proposed to remove sentiment categories that have a relatively small number of samples from the classification scheme. This will result in a more balanced dataset, where the

Type	Size	Emotions	RMSE	RMSE normalized	N Tweets
tfidf	100	neutral	2.120911	0.000246	8638
tfidf	500	worry	3.562678	0.000421	8459
tfidf	500	sadness	2.382576	0.000461	5165
tfidf	500	love	2.158136	0.000562	3842
tfidf	300	happiness	3.550297	0.000682	5209
bow	500	surprise	3.365935	0.001539	2187
bow	100	relief	2.458842	0.001611	1526
tfidf	100	hate	3.265023	0.002468	1323
tfidf	500	fun	4.706005	0.002650	1776
bow	300	enthusiasm	5.584282	0.007357	759
tfidf	100	empty	7.009085	0.008475	827
bow	100	boredom	7.222650	0.040350	179
bow	100	anger	8.168676	0.074261	110

TABLE XI: Best RMSE for each Emotion

	Random Forest classifier		
	Dataset	Dataset_1	Dataset_2
Test-set Accuracy	31.46%	32.42%	31.16%

TABLE XII: Random Forest Classifier

remaining categories are better represented, potentially leading to improved model performance.

For this reason, a new dataset (named 'Dataset_1' in Table (XII) and Table (XIII)) was created from the initial one (named 'Dataset' in Table (XII) and Table(XIII)). In this, the three categories least represented in the initial dataset were removed. This choice was based on the fact that the sum of the tweets of these categories consisted of less than 10% of the number of tweets of the most represented category. At this point, both a Random Forest classifier using 500 TF-IDF size as feature vector and an XGBoost classifier characterized by the same parameters used in 3.D. The accuracy achieved on the test set is shown in Table (XII) and Table (XIII).

In order to further investigate the impact that a change in category design can have on the results, the three categories identified earlier as being the least represented in the initial dataset were merged into a new category called 'meta-category' thus creating a new dataset (called 'Dataset_2 in Table (XII) and Table (XIII)). At this point, the two existing classifiers were utilized and the results are shown in Table (XII) and Table (XIII).

As can be appreciated from the results shown in the tables, there seems to be an increase in performance in terms of accuracy in the case of Dataset_1. This is in agreement with the statements made in the previous point, as it was noted that more represented classes are classified better, thus leading to an increase in total accuracy.

	XGBoost classifier		
	Dataset	Dataset_1	Dataset_2
Test-set Accuracy	33.17%	33.59%	32.95%

TABLE XIII: XGBoost Classifier

IV. Overall Discussions

The dataset used for the project was downloaded from Kaggle and contains 40k tweets. Each tweet is associated with a characterizing sentiment and the aim of the project was an in-depth analysis that touched on several points such as initial exploration of the dataset categorization of the different tweets using Empath LDA and BerTopic. Subsequently, the analysis took into consideration the similarity between tweets belonging to the same category. To do this, FuzzyWuzzy was used and through BERT embedding it was possible to study the relationship between records belonging to a category and the category itself. A similarity analysis was also carried out between different categories (Formula (3)).

As far as the initial exploration of the dataset is concerned, an imbalance between the categories is evident, and this feature of the dataset was also of considerable importance in the last classification task.

As far as categorization is concerned, BerTopic achieved the best results when compared in particular with Empath. LDA also achieved good results considering the limitation to only three required keywords. It is worth mentioning the specific weight of Wordnet. It proved to be a limitation as far as semantic analysis (hypernyms and hyponyms) is concerned.

With regard to the analysis within each category, FuzzyWuzzy showed considerable diversity between Tweets belonging to the same emotion. Speaking of similarity calculated using the Formula (3) that makes use of Wu&Palmer similarity, the results were satisfactory, although, as mentioned above, the lack of antonyms concerning the different categories caused by Wordnet placed a considerable limitation on the analysis (as can be seen from Table (A.XIV)).

In the final analysis, a classification task was conducted using firstly a random forest classifier and subsequently an XGBooster classifier. As was to be expected, XGBoosterclass. achieved better results in terms of accuracy, this being due to its nature being more prone to analyzing large and unbalanced datasets. Next, a random forset class. was implemented by varying the vectorizer types and vector sizes. The results obtained from the different implementations did not show a clear combination of vector size and type of vector that was more perfect than the others. It should be emphasized that this analysis was conducted for each category.

As mentioned above, in order to further investigate the real impact of the imbalance of classes within the dataset, an exploration of the RMSE was carried out by considering the best combination of vector type and size of vector for each category (Table (XI)). The results showed that classes with a higher number of tweets were better classified. Given this conclusion, a modification to the initial dataset was implemented. The three least represented categories were eliminated and the classification of both the random forest class. and the XGBooster class. was repeated.

A second modification to the dataset was made, creating a meta-category containing the three least represented categories. Again, the classification task was performed.

Regarding the latter points, the results (as was to be expected), showed an increase in accuracy, especially in the dataset with only 10 categories. This is due to the lack of the least represented categories, which were consequently classified with a lower accuracy, leading to a general decrease in the accuracy of the entire dataset.

As far as possible future implementations are concerned, it would be interesting to carry out a more in-depth analysis of the categorization using LDA (searching for more than 3 keywords). The comparison between the different categories was strongly influenced by the presence or absence of antonyms within the wordnet. Consequently, an analysis of the level of tweets is recommended. In particular, a semantic analysis could be conducted between tweets belonging to different categories in order to quantify the real diversity between two different categories. Also as future developments, it would be interesting to implement the classification task using a deep learning algorithm [4] that could attempt to achieve better performance despite the fact that the dataset is unbalanced.

Of considerable importance to the eventual development of a deep learning algorithm is the pre-processing carried out in such a dataset. In fact, being a collection of tweets, there are several challenges regarding the nature of people's colloquial language and the syntax of the language used in social networks. For example, the classification of emoji would have a considerable impact on the association between emotion and tweet.

REFERENCES

[1] https://appen.com/pre-labeled-datasets/.

[2] Ethan Fast, Binbin Chen, Michael Bernstein, 'Empath: Understanding Topic Signals in Large-Scale Text', https://doi.org/10.48550/arXiv.1602.06979

[3] Maarten Grootendorst, 'BERTopic: Neural topic modeling with a class-based TF-IDF procedure', https://doi.org/10.48550/arXiv.2203.05794

[4] S. S. Raga and C. B. L, "A bert model for sms and twitter spam ham classification and comparative study of machine learning and deep learning technique," 2022 IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE), MANGALORE, India, 2022, pp. 355-359, doi: 10.1109/ICRAIE56454.2022.10054285.

APPENDIX

	Anger	**Boredom**	**Empty**	**Enthusiasm**	**Fun**	**Happiness**	**Hate**	**Love**	**Neutral**	**Relief**	**Sadness**	**Surprise**	**Worry**
Anger	-	-	-	-	-	-	-	-	-	-	-	-	-
Boredom	-1.00	-	-	-	-	-	-	-	-	-	-	-	-
Empty	0.67	0.67	-	-	-	-	-	-	-	-	-	-	-
Enthusiasm	-1.00	-1.0	0.67	-	-	-	-	-	-	-	-	-	-
Fun	-1.00	-1.0	0.67	-1.0	-	-	-	-	-	-	-	-	-
Happiness	0.56	0.56	0.23	0.56	0.56	-	-	-	-	-	-	-	-
Hate	0.57	0.57	0.24	0.57	0.57	0.13	-	-	-	-	-	-	-
Love	0.57	0.57	0.24	0.57	0.57	0.13	0.14	-	-	-	-	-	-
Neutral	0.75	0.75	0.42	0.75	0.75	0.31	0.32	0.32	-	-	-	-	-
Relief	-1.00	-1.0	0.67	-1.0	-1.0	0.56	0.57	0.57	0.75	-	-	-	-
Sadness	0.58	0.58	0.25	0.58	0.58	0.15	0.15	0.15	0.33	0.58	-	-	-
Surprise	-1.00	-1.0	0.67	-1.0	-1.0	0.56	0.57	0.57	0.75	-1.0	0.58	-	-
Worry	0.83	0.83	0.5	0.83	0.83	0.4	0.4	0.4	0.58	0.83	0.42	0.83	-

TABLE XIV: Emotion Similarity Scores