Kaggle Text to Emotion

1st Abdurrehman Syed
University of Oulu, Computer Science and Engineering
Abdurrehman.Syed@student.oulu.fi 2308546

2nd Antti Korpi
University of Oulu, Computer Science and Engineering
akorpi20@student.oulu.fi Y62761996

3rd Waltteri Rasila
University of Oulu, Computer Science and Engineering
krasila@student.oulu.fi 2463674

***Abstract*—With the advancement of computers and their ability to process and understand Natural Language, countless horizons have been opened in the exciting field of text-to-emotion analysis. This progress has significantly expanded our ability to understand human emotions expressed in the form of textual data, making this report paper, based on text based sentiment analysis, more comprehensive, relevant, and insightful. Since there is large quantities of textual data across multiple digital platforms, such as social media and online forums, it has become important to utilize the potential text-to-emotion analysis has to offer. The purpose of this project is to explore various methodologies and provide detailed explanations and insights into the analysis process.**

***Index Terms*—Text to emotion, NLTK, Python, Transformer, Word2Vec, Gensim, GloVe, BERT, FastText**

I. Introduction

The way we communicate emotions is evolving, with text-based conversations becoming a predominant mode of expression. This has led to a growing interest in the field of text to emotion analysis, where the goal is to interpret the emotional context within the textual content. These text-based sentiment classifications have several use cases including social media insights, trends, content recommendation, customer feedback and many more.

This project highlights the key concepts of sentiment analysis and gives an in-depth overview of the results obtained. With the aim of exploring different approaches, traditional and advanced, compare their outputs, preprocessing and cleaning data as well as observing patterns, and sentiment information in text. Moreover, we merged categories based on certain factors and tested the dataset using Vader and sentiStrength analyzers, compared Wu & Palmer and Cosine similarities between the labels and its records, discussed Word2Vec embeddings, their vectors and explored prominent deep learning models like BERT, GloVe, and FastText.

The implementation is carried out with the help of Python's NLTK (Natural Language Tool Kit) module for (NLP) Natural Language Processing tasks, for data handling, conversions, and normalization the libraries used are Pandas and Numpy. Additionally, Gensim's pretrained Word2Vec model is used for making sentiment predictions, and transformer dependency is used for BERT implementation.

Finally, the results are evaluated and discussed, along with providing additional suggestions, experimentations and recommendations for alternate approaches to handle the task of data analysis.

II. Datasets

The primary dataset used for the analysis is Kaggle's Twitter Dataset, comprising 40,000 records, classified under 13 labels such as happiness, sadness, anger, neutrality, boredom, fun, empty, enthusiasm, hate, love, relief, surprise and worry. The secondary dataset used is NRC word emotion association lexicon which acts as a measure to compare the results with. This lexicon is a list of words of various languages and their association with fundamental emotions. The annotations for both of these datasets were done manually through crowd-sourcing.

III. methodology

In this chapter we explain the problems this project had and how we achieved the specifications each problem had. If any of the problems aren't finished, then we explain what ideas we had to possibly achieve the required specifications.

A. Statistical properties of the data

First step was to initially explore the data and its properties. This was done by downloading the data and writing a small script, that gathers all the tweets associated with certain sentiment. Then the sentiments and their values (amount of tweets per sentiment) were plotted to a single histogram.

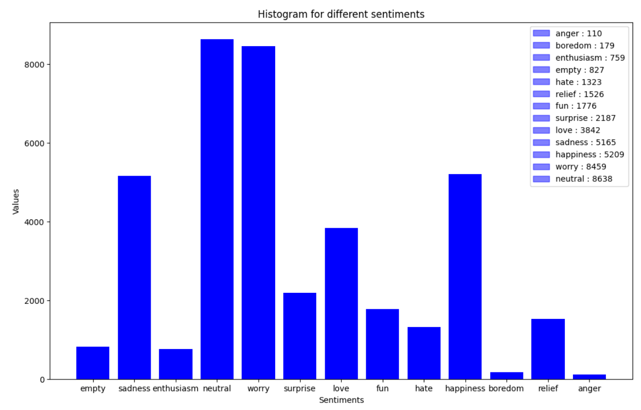

Fig. 1: Histogram of the tweets per sentiment

From the histogram in figure 1 we can see, that the most dominant sentiments are worry and neutral, followed by happiness and sadness. This lets us believe that this dataset of tweets has been collected during a time, when some sort of change was happening, causing people to worry. Perhaps an election is happening for example, making people feel happy, sad, worried or neutral about it. For even better understanding of the data, the amount of tweets per sentiment has also been added in numerical form.

B. Most frequent keywords

Next step included some preprocessing for the data. The problem required us to find the 10 most used keywords for each sentiments before and after processing the data. Preprocessing here included stop word removal and removal of special characters and numbers. Most used keywords for are visible in table I before preprocessing and in table II after preprocessing the data. The percentage values for the keywords are also seen in the tables. It's good to note here that there are still couple stop words remaining even after preprocessing the data, since the stop word list nltk provides, doesn't include stop words when they're written wrong, for example don't written as dont.

Empty	%	Sadness	%	Enthusiasm	%	Anger	%	Worry	%
@	0.31	!	1.68	!	0.4	!	0.06	I	2.86
?	0.2	I	1.67	@	0.29	.	0.03	@	2.83
to	0.2	.	1.66	to	0.22	@	0.03	.	2.8
!	0.19	@	1.61	I	0.21	I	0.03	!	2.79
.	0.18	to	1.45	.	0.2	to	0.02	to	2.43
I	0.16	the	1.18	,	0.18	a	0.02	,	1.9
the	0.16	,	1.12	the	0.15	my	0.02	the	1.88
,	0.14	a	0.83	a	0.13	,	0.02	a	1.53
a	0.11	i	0.83	you	0.11	?	0.02	my	1.27
i	0.09	my	0.8	and	0.1	it	0.02	i	1.25
Hate	%	Fun	%	Love	%	Surprise	%		
!	0.64	!	0.98	!	2.22	!	1.03	-	-
.	0.44	@	0.76	@	1.5	@	0.89	-	-
I	0.42	.	0.57	.	1.08	.	0.62	-	-
@	0.38	,	0.45	I	1	I	0.59	-	-
to	0.35	I	0.45	,	0.88	?	0.59	-	-
the	0.3	to	0.44	to	0.88	,	0.51	-	-
,	0.29	the	0.42	the	0.86	to	0.49	-	-
a	0.24	a	0.34	you	0.71	the	0.46	-	-
i	0.21	and	0.24	a	0.65	a	0.36	-	-
and	0.2	you	0.22	my	0.55	you	0.3	-	-
Bore-dom	%	Relief	%	Happiness	%	Neutral	%	-	-
.	0.07	!	0.56	!	3.32	@	3.54	-	-
to	0.05	.	0.53	@	1.97	.	2.1	-	-
!	0.05	@	0.51	.	1.53	!	1.96	-	-
I	0.04	to	0.41	to	1.28	to	1.94	-	-
the	0.04	I	0.4	,	1.26	I	1.81	-	-
@	0.03	the	0.36	I	1.22	the	1.65	-	-
,	0.02	,	0.35	the	1.2	,	1.64	-	-
...	0.02	a	0.32	a	0.98	?	1.37	-	-
i	0.02	and	0.21	you	0.68	a	1.2	-	-
in	0.02	my	0.21	and	0.68	you	0.96	-	-

TABLE I: Most Frequently occurring words before text-preprocessing

Empty	%	Sadness	%	Enthusiasm	%	Anger	%	Worry	%
im	0.35	im	2.98	im	0.31	im	0.04	im	4.51
dont	0.2	miss	1.49	go	0.22	dont	0.03	dont	2.38
get	0.17	get	1.33	good	0.19	get	0.03	get	2.38
go	0.14	go	1.3	get	0.19	go	0.02	like	2.01
day	0.12	day	1.25	want	0.19	like	0.02	cant	1.92
got	0.12	like	1.21	day	0.16	got	0.02	got	1.86
like	0.12	work	1.17	like	0.14	good	0.022	go	1.85
bored	0.10	dont	1.10	got	0.14	know	0.02	going	1.53
need	0.10	sad	1.10	new	0.14	going	0.02	know	1.52
one	0.10	cant	1.09	work	0.13	work	0.02	day	1.47
Hate	%	Fun	%	Boredom	%	Surprise	%		
hate	0.91	im	0.73	im	0.15	im	0.83	-	-
im	0.56	lol	0.48	bored	0.08	get	0.52	-	-
like	0.41	like	0.44	work	0.06	day	0.47	-	-
dont	0.38	fun	0.43	really	0.05	got	0.42	-	-
get	0.34	good	0.42	go	0.05	like	0.41	-	-
work	0.31	get	0.39	like	0.05	know	0.41	-	-
cant	0.29	go	0.38	amp	0.04	cant	0.40	-	-
really	0.26	day	0.35	dont	0.04	oh	0.40	-	-
got	0.23	u	0.34	still	0.04	dont	0.39	-	-
go	0.22	see	0.33	getting	0.04	see	0.38	-	-
Love	%	Relief	%	Happiness	%	Neutral	%	-	-
love	3.64	im	0.61	im	2.08	im	2.48	-	-
day	2.65	day	0.57	day	2.01	get	1.62	-	-
mothers	2.46	good	0.49	good	1.94	go	1.49	-	-
happy	1.45	got	0.38	happy	1.21	like	1.46	-	-
im	1.25	thanks	0.35	great	1.17	dont	1.26	-	-
good	1.15	like	0.35	got	1.16	good	1.24	-	-
like	0.78	get	0.30	thanks	1.0	day	1.23	-	-
u	0.76	back	0.28	like	1.02	got	1.23	-	-
thanks	0.70	dont	0.27	get	0.96	one	1.14	-	-
great	0.58	time	0.27	mothers	0.90	know	1.09	-	-

TABLE II: Most Frequently occurring words after text-preprocessing

From the tables I and II we can see words associated with the correct sentiments, such as sadness containing words miss and sad, or enthusiasm containing words good and day. Initially we guessed the tweet data set could've been collected during a time of election, but seeing the 10 most used keywords for each sentiments certainly seems otherwise. Figures are in a form of pandas dataframe, since plotting these to histograms would've given us 20 different graphs, or one very messy graph.

C. Part-of-speech taggers

For this step, two types of Part of Speech (PoS) taggers were required. Perceptron tagger which is trained on 45 Penn Treebank's tags and Unigram tagger that treats each word in the text separately are applied to every dataframe, among which the top five most tagged tags are shown as their outputs for each category where the first element is the tag and the second is its occurrence represented as a percentage. These PoS taggers were opted as they provide different views of the same data, giving more insight and explanation.

For implementation, Penn Treebank was imported from NLTK's corpus module along with Perceptron and Unigram taggers, in addition to that, the Unigram tagger was first trained on Treebank's golden tagset and later used for making predictions shown in table III, whereas table IV represents the results of Perceptron tagger.

Empty	%	Sadness	%	Enthusiasm	%	Anger	%
None	28.7	None	26.7	None	26.6	None	27.2
IN	9.92	IN	9.17	IN	9.19	IN	9.60
.	7.09	NN	6.42	.	6.4	.	7.07
NN	6.06	PRP	6.22	NN	6.39	NN	6.34
PRP	5.33	.	6.21	PRP	6.22	PRP	6.09
Neutral	%	Worry	%	Surprise	%	-	-
None	27.4	None	26.6	None	26.6	-	-
IN	10.1	IN	9.77	IN	9.74	-	-
.	6.38	NN	6.40	.	6.58	-	-
NN	6.35	.	6.39	NN	6.35	-	-
PRP	5.98	PRP	6.19	PRP	6.21	-	-
						-	-
Love	%	Fun	%	Hate	%	-	-
None	26.9	None	27.1	None	27.1	-	-
IN	9.63	IN	9.64	IN	9.60	-	-
.	6.76	.	6.82	.	6.86	-	-
NN	6.30	NN	6.29	NN	6.29	-	-
PRP	6.22	PRP	6.18	PRP	6.18	-	-
						-	-
Happi-ness	%	Boredom	%	Relief	%	-	-
None	27.3	None	27.3	None	27.2	-	-
IN	9.59	IN	9.59	IN	9.60	-	-
.	7.09	.	7.09	.	7.07	-	-
NN	6.30	NN	6.30	NN	6.34	-	-
PRP	6.09	PRP	6.09	PRP	6.08	-	-

TABLE III: PoS tagging with Unigram Tagger

D. WordNet Lexical Database

NLTK's WordNet lexicon is used as a standard to compare the words presented in each tweet of every category with the database's occurrence of that word in different contexts, every unmatched and matched tweet is recorded and summarized as a percentage of the unmatched records in a table.

E. Stylometric Features

NRC Word Emotion Association Lexicon (EmoLex) is downloaded from GitHub and used in this project as a measure to compare the lexical vocabulary of tweets and their sentiments with the vocabulary in EmoLex. A Boolean variable 'm' is used to check the individual words for each record in Twitter dataset and compare its occurrence in the same label class in the NRC lexicon. For instance, a word 'jolly' in a tweet with label category as happy is compared with the same label category in EmoLex and the Boolean variable is set to one only if both categories have that word otherwise its set

Empty	%	Sadness	%	Enthusiasm	%	Anger	%
NN	16.9	NN	16.1	NN	15.9	NN	16.3
NNP	8.76	JJ	7.93	JJ	7.88	NNP	8.39
JJ	7.33	NNP	7.37	NNP	7.58	JJ	7.96
.	7.09	RB	6.5	IN	6.42	.	7.07
IN	6.38	IN	6.47	.	6.4	IN	6.45
Neutral	%	Worry	%	Surprise	%	-	-
NN	16.4	NN	16.2	NN	16.13	-	-
NNP	8.60	NNP	7.96	NNP	8.02	-	-
JJ	7.67	JJ	7.74	JJ	7.75	-	-
IN	6.58	IN	6.57	.	6.58	-	-
.	6.37	.	6.39	IN	6.52	-	-
						-	-
Love	%	Fun	%	Hate	%	-	-
NN	16.1	NN	16.2	NN	16.2	-	-
NNP	8.28	NNP	8.34	NNP	8.28	-	-
JJ	7.87	JJ	7.87	JJ	7.86	-	-
.	6.76	.	6.82	.	6.86	-	-
IN	6.42	IN	6.41	IN	6.42	-	-
						-	-
Happi-ness	%	Boredom	%	Relief	%	-	-
NN	16.3	NN	16.3	NN	16.3	-	-
NNP	8.44	NNP	8.43	NNP	8.40	-	-
JJ	7.97	JJ	7.96	JJ	7.95	-	-
.	7.09	.	7.09	.	7.07	-	-
IN	6.41	IN	6.41	IN	6.45	-	-

TABLE IV: PoS tagging with Perceptron Tagger

to zero. Finally, the Pearson Correlations and P-Values are calculated for each category with the help of SciPy's stats module.

Pearson Correlation measures the linear relationship between two continuous variables. It calculates how well a variation in one variable can be predicted by the variation in the other variable. The mathematical representation is as follows:

$$r = \frac{\sum_{i=1}^{N}((X_i - \overline{X})(Y_i - \overline{Y}))}{\sqrt{\sum_{i=1}^{N}(X_i - \overline{X})^2 \sum_{i=1}^{N}(Y_i - \overline{Y})^2}} \tag{1}$$

Categories	Identified	Unidentified	Percentage
Empty	80	39	32.773
Sadness	82	44	34.921
Enthusiasm	78	38	32.759
Neutral	85	50	37.037
Worry	85	50	37.037
Surprise	83	43	34.127
Love	82	45	35.433
Fun	80	43	34.959
Hate	82	39	32.231
Happiness	84	46	35.385
Boredom	68	31	31.313
Relief	82	44	34.921
Anger	56	32	36.364

Where:

$$N\ (Sample) = the\ number\ of\ data\ points$$
$$X_i\ and\ Y_i = individual\ points$$
$$\overline{X}\ and\ \overline{Y} = datasets\ x\ and\ y$$

Categories	Pearson Correlation	P-value
Empty	0.592	1.872e-79
Sadness	0.452	3.9e-259
Enthusiasm	0.473	1.196e-43
Neutral	0.571	0
Worry	0.471	0
Surprise	0.522	1.729-e153
Love	0.451	9.68-e193
Fun	0.46	7.88e-94
Hate	0.396	3.78e-51
Happiness	0.468	4.366e-282
Boredom	0.332	0.000005
Relief	0.477	6.5e-88
Anger	0.53	2.421e-09

TABLE V: Pearson Correlations and P-Values

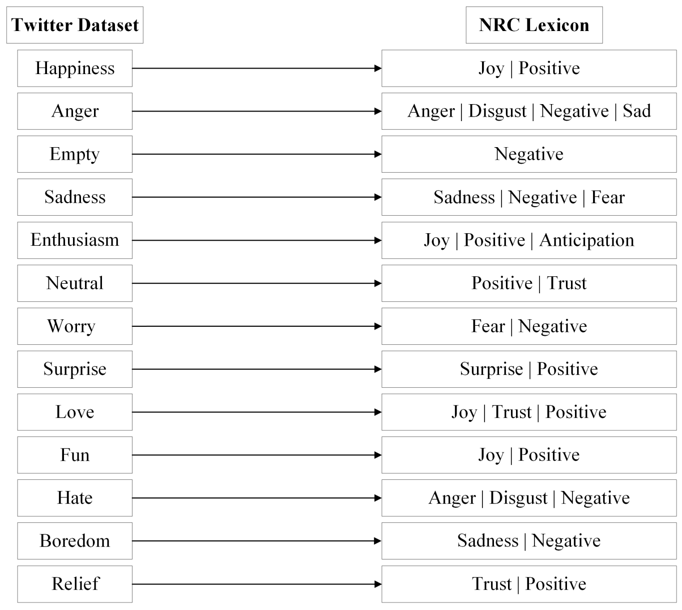

Fig. 2: Kaggle and NRC datasets mapping

F. *Vader Sentiment Analyzer*

VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analyzer is a lexicon and rule-based sentiment analysis tool designed for text based sentiment analysis, with a particular focus on social media content. It assigns polarity scores to individual words based on the scores it's database has for that particular word. Afterwards, it calculates and returns the compound polarity for the entire sentence.

In our project we analyzed the labels for each category as to which sentiment class do they belong (negative, positive or neutral), for this reason "Vader" is used which outputs a 3x1 tuple representing the sentiment classes. The vader analyzer was used for finding out the sentiment score for each categorical label and subsequently, for each individual record within that category, the outputs were compared as how many tweets actually belong to that particular category according to the analyzer and how many do not. Followed by that, the results were stored in a Json file and shown as a percentage of the words that matched the category and words that did not match in a tabular representation.

G. *SentiStrength Sentiment Analyzer*

SentiStrength is another sentiment analysis tool, but it operates differently from VADER. SentiStrength uses a combination of lexicon-based and rule-based approaches. It's designed to analyze text for sentiment, specifically focusing on the strength or intensity of positive and negative emotion in the text.

In the analyzer the sentiment scoring works in a way that when a piece of text is analyzed, each of its individual words are looked and compared with SentiStrength's own lexical database, if a word is present it will be assigned a score. For

Categories	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Incorrectly Labeled	588	2530	331	4847	4980	1170	653	591	458	1212	85	557	61
Correctly Labeled	239	2635	428	3791	3479	1017	3189	1185	865	3997	94	969	49
Total Samples	827	5165	759	8638	8459	2187	3842	1776	1323	5209	179	1526	110

TABLE VI: VADER Analyzer

Categories	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Correctly Labeled	360	2733	422	4018	3616	577	3091	1160	908	4046	97	881	53
Incorrectly Labeled	467	2432	337	4620	4843	1610	751	616	415	1163	82	645	57
Total Samples	827	5165	759	8638	8459	2187	3842	1776	1323	5209	179	1526	110

TABLE VII: SentiStrength Analyzer

instance, ’happy’ might get a positive score while ’angry’ get a negative score.

However, SentiStrength doesn’t limit itself with only the word-matching approach, instead, it employs a rule based technique where words like ’very happy’ are assigned higher scores than simply ’happy’ and negations like ’not happy’ are taken into account.

In our project we passed the exact same list of categorical labels and tweet records as we did for the VADER analyzer, and got a similar form of output, a list with three values indicating positive, negative and neutral emotions. But the matched and unmatched scores were different than VADER. As shown in the tables VI and VII

H. Handling Unbalanced Classes With Wu & Palmer Similarity

To reduce the bias representation of data, an approach was suggested to merge classes that classify in similar emotions such as, ‘anger’ and ‘hate’. For this reason, based on the data exploration conducted in the earlier tasks, a threshold of 1,500 frequencies of tweets is set, anything below this number is considered a small category and are up for merging with other relevant categories or with themselves, following are the categories that pass this threshold *empty, enthusiasm, hate, boredom, and anger*. For effective implementation of this methodology the Wu and Palmer similarities are calculated between each of the 13 categories in the Kaggle Twitter dataset. Their similarities are presented in Table VIII.

The helper module of Itertools is used to find all possible combinations between them and wordnet’s synonym sets are used to find all possible use-cases and compare the similarities between the different possible lemmas, out of which the biggest similarity is returned along with their label pair. Afterwards the merging categories are selected based on the label that has the most matching pairs, if two or more labels have equal number of matching pairs then their average similarity between all other lemmas of that pair is taken into consideration and based off that a category is selected to merge the smaller ones with.

I. Handling Unbalanced Classes With Cosine Similarity

In the previous topic heading, we addressed the issue of handling unbalanced classes by exploring the Wu and Palmer semantic similarity between categories. This approach allowed us to identify potential candidates for merging categories with relatively small training sample sizes. Building on that foundation, we now explore a different approach that leverages the power of Word2Vec and cosine similarity to achieve the same goal but with different results.

Word2Vec is a popular technique in natural language processing (NLP) that’s used for word embedding, representing words as dense vectors in a multi-dimensional space. These vectors capture semantic relationships between words. In our project we implemented the Word2Vec technique with the help of Gensim module’s pre-trained model. Gensim is a Python library that provides tools for working with Word2Vec models. By calling it, we can harness the power of pre-trained Word2Vec models, or even train our own models on specific datasets. The result is a 100 dimensional word embedding that enables us to measure the semantic similarity between words or phrases using cosine similarity.

We followed the exact same methodology as previous handling technique, however, this time, we did it through the lens of Word2Vec and cosine similarity. The threshold for possible merging categories still remained 1,500 but for their cosine similarity score it was set to 0.1. Further, we calculated the similarity score between every possible label pair as shown

	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Empty	1	0.25	0.222	0.5	0.5	0.4	0.5	0.222	0.4	0.222	0.153	0.555	0.4
Sadness	0.25	1	0.833	0.25	0.714	0.769	0.769	0.545	0.769	0.833	0.625	0.714	0.769
Enthusiasm	0.222	0.833	1	0.222	0.714	0.769	0.769	0.571	0.769	0.833	0.625	0.714	0.769
Neutral	0.5	0.25	0.222	1	0.5	0.4	0.705	0.222	0.4	0.222	0.153	0.705	0.4
Worry	0.5	0.714	0.714	0.5	1	0.666	0.8	0.428	0.8	0.75	0.555	0.625	0.8
Surprise	0.4	0.769	0.769	0.4	0.666	1	0.714	0.571	0.714	0.769	0.588	0.857	0.714
Love	0.5	0.769	0.769	0.705	0.8	0.714	1	0.461	0.857	0.8	0.666	0.666	0.857
Fun	0.222	0.545	0.571	0.222	0.428	0.571	0.461	1	0.451	0.5	0.375	0.8	0.666
Hate	0.4	0.769	0.769	0.4	0.8	0.714	0.857	0.461	1	0.8	0.588	0.666	0.857
Happiness	0.222	0.833	0.833	0.222	0.75	0.769	0.8	0.5	0.8	1	0.625	0.714	0.8
Boredom	0.153	0.625	0.625	0.153	0.555	0.588	0.666	0.375	0.588	0.625	1	0.555	0.588
Relief	0.555	0.714	0.714	0.705	0.625	0.857	0.666	0.8	0.666	0.714	0.555	1	0.666
Anger	0.4	0.769	0.769	0.4	0.8	0.714	0.857	0.666	0.857	0.8	0.588	0.666	1

TABLE VIII: Wu & Palmer Similarities

in Table IX which also highlights the possible semantically similar classes for merging.

Lastly, the columns to be merged are joined with a unique name that takes into account of each of the prior names of the categories and drops their own separate columns.

J. Enhancing Category Matching with Cosine Similarity

In the preceding sections, we addressed the crucial task of matching the similarity between categories' titles, however, for this section we went a step further and now calculate the embedding vectors for each record of those title categories. Similar to the label-matching the goal is to now discuss the similarities of the records with their categories and their similarity with other records from other categories. To achieve this, we employ a method that combines Word2Vec embedding, cosine similarity, and extensive data analysis.

Our methodology begins by defining a function that receives two categories, *k1* and *k2*, as input and calculates the similarity between the records within these categories. This function capitalizes on Word2Vec embeddings, computing the embedding for each record as the average of the embeddings of the words it contains. This step ensures that the content of each record is accurately represented in the vector space. In other words, if a tweet contains a text like: *"Feeling hungry, may order cheese pizza tonight"*, what the function would do is that it will calculate the vector embeddings for each word in this tweet and then average all the resulting embeddings to get a single vector representation of the entire tweet.

Additionally, the subsequent stage involves generating embeddings for the overall list of records in a category rather than the aforementioned method which generates the embeddings for each and every word. This is accomplished by calculating the mean of the embedding vectors associated with each individual record in the category. The resulting embeddings encapsulate the collective semantic representation of records in a category.

To measure the record-based category similarity, we employ cosine similarity. It quantifies the similarity between two embedding vectors, proving to be a reliable metric to measure the embeddings with. To assess the alignment between these embeddings, we further create a table, shown in Table X, which highlights the relationship and similarities.

At last, a method *record_based_similarity_between_each_pair* was created which calculates both word and list based Word2Vec embeddings for each possible combination of records based on their labeled categories and then calculates the cosine similarities between them and finally displaying the results.

K. Exploration Of Modern Techniques

So far we have been working with Word2Vec and other traditional approaches, however, in this section we will encircle some modern day techniques that are very powerful and efficient. The task performed in previous section will also be performed with these methodologies but the results will differ.

1) GloVe (Global Vectors for Word Representation): GloVe is an unsupervised learning algorithm which is used for obtaining vector representations of words. It uses a global context approach and considers the entire corpus of text to learn word embeddings. GloVe starts by constructing a global word-word co-occurrence matrix from the entire text lexicon where the elements of this matrix represent how often words

	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Empty	1	-0.03	0.128	-0.045	-0.04	-0.004	0.0007	-0.145	0.15	-0.009	0.046	-0.003	0.093
Sadness	-0.03	1	0.109	-0.074	0.037	0.111	-0.083	0.166	-0.005	0.041	0.172	0.008	0.079
Enthusiasm	0.128	0.109	1	0.015	0.026	0.108	0.05	0.02	-0.25	0.012	-0.101	-0.134	0.062
Neutral	-0.045	-0.074	0.015	1	0.044	-0.093	0.001	-0.105	-0.173	-0.169	-0.032	-0.095	0.216
Worry	-0.04	0.037	0.026	0.044	1	-0.144	0.008	0.06	-0.069	0.074	0.199	0.03	0.0227
Surprise	-0.04	0.111	0.108	-0.09	-0.144	1	0.004	-0.057	-0.028	-0.013	-0.002	-0.115	0.092
Love	0.007	-0.083	0.05	0.001	0.008	0.004	1	0.019	0.034	0.041	0.145	-0.114	0.016
Fun	-0.145	0.166	0.02	-0.105	0.06	-0.057	0.019	1	0.138	0.131	0.064	0.009	-0.059
Hate	0.15	-0.005	-0.258	-0.173	-0.069	-0.028	0.034	0.138	1	-0.044	0.17	0.004	-0.027
Happiness	-0.009	0.041	0.012	-0.169	0.074	-0.013	0.041	0.131	-0.044	1	-0.013	0.067	-0.111
Boredom	0.046	0.172	-0.101	-0.032	0.199	-0.002	0.145	0.064	0.17	-0.013	1	-0.023	-0.052
Relief	-0.003	0.008	-0.134	-0.095	0.033	-0.115	-0.114	0.009	0.004	0.067	-0.023	1	-0.01
Anger	0.093	0.079	0.062	0.216	0.027	0.092	0.016	-0.059	-0.0277	-0.111	-0.052	-0.01	1

TABLE IX: Cosine Similarities

	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Empty	0.686	0.641	0.638	0.65	0.645	0.632	0.534	0.623	0.622	0.534	0.817	0.609	0.904
Sadness	0.71	0.666	0.669	0.675	0.671	0.662	0.561	0.644	0.651	0.56	0.826	0.632	0.9
Enthusiasm	0.71	0.665	0.672	0.657	0.671	0.662	0.562	0.636	0.652	0.563	0.823	0.63	0.895
Neutral	0.685	0.636	0.635	0.647	0.64	0.631	0.529	0.622	0.618	0.529	0.812	0.607	0.902
Worry	0.712	0.667	0.673	0.676	0.673	0.666	0.564	0.645	0.653	0.565	0.825	0.632	0.898
Surprise	0.711	0.673	0.672	0.674	0.674	0.662	0.565	0.646	0.654	0.564	0.828	0.633	0.898
Love	0.709	0.665	0.671	0.67	0.672	0.66	0.564	0.643	0.652	0.565	0.824	0.631	0.897
Fun	0.724	0.683	0.69	0.69	0.69	0.676	0.582	0.644	0.671	0.581	0.832	0.644	0.894
Hate	0.712	0.671	0.673	0.676	0.696	0.661	0.566	0.647	0.655	0.566	0.832	0.634	0.903
Happiness	0.712	0.67	0.675	0.675	0.675	0.664	0.566	0.646	0.655	0.567	0.828	0.634	0.898
Boredom	0.703	0.656	0.661	0.666	0.663	0.655	0.554	0.637	0.64	0.555	0.821	0.624	0.898
Relief	0.711	0.667	0.673	0.69	0.673	0.663	0.563	0.645	0.653	0.564	0.824	0.631	0.897
Anger	0.708	0.668	0.668	0.671	0.669	0.657	0.563	0.63	0.65	0.562	0.832	0.631	0.904

TABLE X: Word & List Based Similarities

co-occur in the same context. In our project we installed and used GloVe's Twitter model, *glove.twitter.27B.100d.txt* to be more precise. This means that the pre-trained model used is trained on twitter's language or tweets and understands the context more in a global view. Due to this reason we can expect the model to perform well and give decent results.

The methodology remains quite similar to the previous task, however, in this task only the word-based similarities were calculated as GloVe vectors are implemented on words and not on lists.

2) FastText: FastText is a popular open-source, free, lightweight library developed by Facebook's AI Research lab for text classification and word representation. An extension of the Word2Vec model and is designed to efficiently learn

	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Empty	1	0.997	0.998	0.999	0.997	0.998	0.99	0.997	0.995	0.993	0.994	0.996	0.996
Sadness	0.997	1	0.996	0.996	1	0.996	0.99	0.996	0.998	0.993	0.997	0.998	0.998
Enthusiasm	0.998	0.996	1	0.999	0.997	0.999	0.995	0.999	0.994	0.998	0.992	0.998	0.996
Neutral	0.999	0.996	0.999	1	0.997	0.998	0.993	0.998	0.994	0.996	0.992	0.997	0.995
Worry	0.997	1	0.997	0.997	1	0.997	0.989	0.996	0.998	0.993	0.997	0.998	0.998
Surprise	0.998	0.996	0.999	0.998	0.997	1	0.995	0.999	0.994	0.997	0.991	0.997	0.996
Love	0.99	0.99	0.995	0.993	0.989	0.995	1	0.996	0.985	0.998	0.982	0.993	0.988
Fun	0.997	0.996	0.999	0.998	0.996	0.999	0.996	1	0.993	0.999	0.991	0.997	0.995
Hate	0.995	0.998	0.994	0.994	0.998	0.994	0.985	0.993	1	0.989	0.997	0.995	0.998
Happiness	0.993	0.993	0.998	0.996	0.993	0.997	0.998	0.999	0.989	1	0.987	0.997	0.992
Boredom	0.994	0.997	0.992	0.992	0.997	0.991	0.982	0.991	0.997	0.987	1	0.995	0.996
Relief	0.996	0.998	0.998	0.997	0.998	0.997	0.993	0.997	0.995	0.997	0.995	1	0.996
Anger	0.996	0.998	0.996	0.995	0.998	0.996	0.988	0.995	0.998	0.992	0.996	0.996	1

TABLE XI: GloVe Embeddings Based Similarities

word representations and perform text classification tasks.

In our project we used FastText for calculating similarities such as we did in the prior methods, for this purpose we used Gensim's FastText pre-trained model implementation.

3) BERT (Bidirectional Encoder Representations from Transformers): A powerful and widely used NLP model developed by Google AI. Known for its ability to understand the context of words in a sentence by considering both the left and right context, unlike earlier models that only looked in one direction.

In this project BERT was used to calculate similarities between records of different categories. It was implemented through the help of Transformer library by importing the model along with its tokenizer, passing the tweet data and getting tensors as output. Further, base on these tensors the cosine similarities was calculated.

L. Extra State Of Art Approaches

After working and understanding the prior models, techniques and architectures, we delve deeper into Transformers, Hugging Face documentation to be precise and other language models such as GPT2 testing and exploring them.

1) RoBERTa (A Robustly Optimized BERT Pretraining Approach): RoBERTa is a variant of the Transformer-based language model originally introduced by OpenAI. It was designed to improve upon BERT, some of the key improvements are: larger scale training, use of dynamic masking, and no sentence order prediction. Its performance similarity can be observed in Table XIV.

2) DistilBERT: DistilBERT is another variant of BERT which is designed to be smaller and faster while retaining the performance of BERT model. It reduces the numbers of parameters resulting in a faster performance.

In our project we have implemented it in the same way as BERT by importing the model from transformers module along with its tokenizer, however, the results it gave were quite astonishing.

3) GPT-2 (Generative Pre-trained Transformer 2): GPT-2, a successor of GPT-1 and a very powerful and advanced learning model. We used it in our project to see a rough estimate of how it will predict the similarities. We imported it in the same way as prior models, through Transformer library along with its tokenizer, the word and list based records were passed to it, GPT-2 created vectors from them and calculated the similarities which are highlighted in the Table XVI.

M. Extra Papers Researched

One method created to extract emotion from text is LRA-DNN (Leaky Relu activated Deep Neural Network). This model has four different categories: pre-processing, feature extraction, ranking and classification. When the results achieved by this method were compared to other state-of-the-art methods by using datasets publicly available, it was found that LRA-DNN achieved "highest accuracy, sensitivity, and specificity" with the results of 94.77%, 92.23%, and 95.91%.

In pre-processing, undesired social comments are removed before the tokenization is carried out for the input text. Tokenization is followed by punctuation and stop word removal and finally lemmatization of the data is implemented. Feature extraction includes extraction of both emoticon and non-emoticon features. These features are then ranked respectively. Both emoticon and non-emoticon features have seven different rank values and classes they're classified under.

	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Empty	1	0.997	0.998	0.999	0.997	0.998	0.99	0.997	0.995	0.993	0.994	0.996	0.996
Sadness	0.997	1	0.996	0.996	1	0.996	0.99	0.996	0.998	0.993	0.997	0.998	0.998
Enthusiasm	0.998	0.996	1	0.999	0.997	0.999	0.995	0.999	0.994	0.998	0.992	0.998	0.996
Neutral	0.999	0.996	0.999	1	0.997	0.998	0.993	0.998	0.994	0.996	0.992	0.997	0.995
Worry	0.997	1	0.997	0.997	1	0.997	0.989	0.996	0.998	0.993	0.997	0.998	0.998
Surprise	0.998	0.996	0.999	0.998	0.997	1	0.995	0.999	0.994	0.997	0.991	0.997	0.996
Love	0.99	0.99	0.995	0.993	0.989	0.995	1	0.996	0.985	0.998	0.982	0.993	0.988
Fun	0.997	0.996	0.999	0.998	0.996	0.999	0.996	1	0.993	0.999	0.991	0.997	0.995
Hate	0.995	0.998	0.994	0.994	0.998	0.994	0.985	0.993	1	0.989	0.997	0.995	0.998
Happiness	0.993	0.993	0.998	0.996	0.993	0.997	0.998	0.999	0.989	1	0.987	0.997	0.992
Boredom	0.994	0.997	0.992	0.992	0.997	0.991	0.982	0.991	0.997	0.987	1	0.995	0.996
Relief	0.996	0.998	0.998	0.997	0.998	0.997	0.993	0.997	0.995	0.997	0.995	1	0.996
Anger	0.996	0.998	0.996	0.995	0.998	0.996	0.988	0.995	0.998	0.992	0.996	0.996	1

TABLE XII: FastText Embeddings Based Similarities

	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Empty	0.89	0.864	0.881	0.859	0.869	0.882	0.889	0.905	0.862	0.888	0.869	0.863	0.871
Sadness	0.886	0.876	0.884	0.859	0.875	0.886	0.899	0.911	0.869	0.894	0.871	0.866	0.876
Enthusiasm	0.89	0.874	0.901	0.869	0.88	0.893	0.906	0.922	0.875	0.907	0.875	0.876	0.883
Neutral	0.897	0.877	0.896	0.884	0.878	0.901	0.912	0.926	0.879	0.911	0.876	0.877	0.889
Worry	0.881	0.867	0.881	0.853	0.877	0.881	0.89	0.905	0.866	0.887	0.868	0.862	0.873
Surprise	0.889	0.872	0.89	0.871	0.875	0.9	0.906	0.92	0.875	0.904	0.871	0.868	0.883
Love	0.883	0.873	0.891	0.869	0.871	0.893	0.917	0.924	0.871	0.908	0.865	0.868	0.88
Fun	0.892	0.879	0.9	0.878	0.879	0.901	0.918	0.935	0.881	0.916	0.874	0.879	0.889
Hate	0.886	0.874	0.888	0.864	0.878	0.893	0.901	0.916	0.884	0.899	0.872	0.866	0.884
Happiness	0.894	0.881	0.902	0.88	0.881	0.903	0.92	0.933	0.882	0.922	0.877	0.882	0.89
Boredom	0.886	0.866	0.882	0.856	0.871	0.882	0.891	0.907	0.866	0.89	0.874	0.864	0.873
Relief	0.895	0.878	0.896	0.869	0.881	0.892	0.904	0.921	0.874	0.906	0.879	0.883	0.883
Anger	0.891	0.876	0.892	0.872	0.88	0.896	0.906	0.921	0.881	0.905	0.874	0.872	0.891

TABLE XIII: BERT Embeddings Based Similarities

Unlike our research, which was done for 13 different sentiments, emotion detection for LRA-DNN works with seven basic emotions: anger, disgust, fear, guilt, joy, sadness, and shame. [1]

IV. Results Discussion & Evaluation

The distribution in Figure 1 showed that out of 13 categories, 4 of them had a tweet frequency between 0-1000, 3 of them ranged from 1000-2000, 1 category in 2000-3000 and 1 in 3000-4000 range, 2 occurrences were found between 5000-6000 frequencies and lastly 2 were present in 8000-9000 number. This spread highlights several important pieces of information. Firstly, the diversity in emotions, the distribution of emotions which may present that people tend to tweet more when they are feeling certain strong emotions

	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Empty	0.932	0.92	0.927	0.914	0.923	0.928	0.933	0.94	0.913	0.93	0.919	0.919	0.919
Sadness	0.928	0.927	0.927	0.912	0.926	0.929	0.939	0.942	0.916	0.931	0.919	0.918	0.921
Enthusiasm	0.935	0.927	0.944	0.923	0.931	0.938	0.947	0.953	0.923	0.945	0.925	0.928	0.928
Neutral	0.938	0.928	0.938	0.932	0.928	0.942	0.948	0.954	0.925	0.945	0.925	0.927	0.932
Worry	0.926	0.921	0.926	0.909	0.928	0.927	0.934	0.94	0.915	0.928	0.917	0.916	0.919
Surprise	0.932	0.925	0.934	0.923	0.927	0.942	0.945	0.952	0.924	0.942	0.921	0.922	0.929
Love	0.929	0.927	0.936	0.922	0.926	0.954	0.938	0.955	0.922	0.945	0.918	0.923	0.927
Fun	0.938	0.932	0.943	0.93	0.933	0.964	0.956	0.964	0.93	0.953	0.927	0.932	0.935
Hate	0.933	0.929	0.935	0.921	0.932	0.94	0.945	0.951	0.934	0.941	0.925	0.923	0.933
Happiness	0.939	0.933	0.945	0.932	0.933	0.947	0.956	0.963	0.931	0.956	0.928	0.934	0.936
Boredom	0.931	0.923	0.93	0.914	0.926	0.931	0.936	0.944	0.919	0.933	0.924	0.921	0.922
Relief	0.933	0.924	0.935	0.918	0.927	0.932	0.94	0.948	0.917	0.939	0.923	0.928	0.923
Anger	0.936	0.931	0.937	0.926	0.933	0.942	0.948	0.954	0.931	0.945	0.926	0.927	0.937

TABLE XIV: RoBERTa Embeddings Based Similarities

Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger	
Empty	0.987	0.988	0.99	0.989	0.989	0.992	0.99	0.994	0.988	0.991	0.988	0.99	0.989
Sadness	0.983	0.986	0.987	0.986	0.986	0.99	0.987	0.993	0.985	0.989	0.984	0.987	0.986
Enthusiasm	0.984	0.986	0.988	0.986	0.986	0.99	0.987	0.993	0.985	0.99	0.985	0.987	0.986
Neutral	0.987	0.988	0.99	0.99	0.989	0.992	0.99	0.994	0.988	0.991	0.987	0.99	0.989
Worry	0.981	0.984	0.985	0.984	0.985	0.989	0.986	0.991	0.983	0.988	0.983	0.985	0.985
Surprise	0.984	0.986	0.988	0.987	0.987	0.991	0.988	0.993	0.986	0.99	0.985	0.988	0.987
Love	0.981	0.984	0.985	0.986	0.984	0.99	0.987	0.992	0.983	0.989	0.982	0.985	0.985
Fun	0.981	0.984	0.986	0.986	0.984	0.993	0.983	0.989	0.985	0.989	0.982	0.986	0.985
Hate	0.983	0.985	0.986	0.986	0.986	0.99	0.987	0.992	0.985	0.989	0.984	0.986	0.986
Happiness	0.982	0.985	0.987	0.987	0.986	0.991	0.988	0.993	0.985	0.99	0.984	0.987	0.986
Boredom	0.985	0.987	0.989	0.987	0.988	0.991	0.988	0.993	0.987	0.99	0.986	0.988	0.988
Relief	0.984	0.986	0.988	0.987	0.987	0.99	0.988	0.993	0.986	0.99	0.985	0.988	0.987
Anger	0.985	0.987	0.989	0.988	0.988	0.991	0.989	0.994	0.987	0.991	0.986	0.988	0.988

TABLE XV: DistilBERT Embeddings Based Similarities

like happiness, worry, and neutrality. This also shows the imbalance in categories as some of them have very large amounts of data compared to others, which will lead to biased and false positive results.

Initially in Table I and Table II it has been observed that prior to pre-processing the most occurring words had no potential meaning to them, whereas, after removing them we can see some hints of relevant occurring words that may help us in recognizing the category to which they belong to. For instance, in Fig 2 the occurrence of the words ‘miss’ in sadness and ‘good’ in enthusiasm gives some context on their root categories.

The PoS tagging process helps in understanding the structure and semantics of the sentences in the tweets. For example,

	Empty	Sadness	Enthusiasm	Neutral	Worry	Surprise	Love	Fun	Hate	Happiness	Boredom	Relief	Anger
Empty	0.986	0.987	0.99	0.987	0.988	0.99	0.989	0.993	0.988	0.991	0.987	0.989	0.987
Sadness	0.986	0.986	0.99	0.986	0.987	0.99	0.988	0.993	0.987	0.991	0.987	0.988	0.987
Enthusiasm	0.985	0.986	0.989	0.985	0.987	0.989	0.988	0.992	0.987	0.99	0.986	0.987	0.986
Neutral	0.988	0.989	0.992	0.989	0.99	0.992	0.991	0.995	0.99	0.993	0.989	0.99	0.989
Worry	0.985	0.985	0.989	0.985	0.987	0.989	0.987	0.992	0.986	0.99	0.986	0.987	0.986
Surprise	0.987	0.988	0.991	0.988	0.989	0.991	0.99	0.989	0.989	0.992	0.988	0.99	0.988
Love	0.987	0.988	0.991	0.987	0.989	0.991	0.994	0.991	0.988	0.992	0.988	0.989	0.988
Fun	0.987	0.987	0.99	0.987	0.988	0.99	0.989	0.989	0.994	0.991	0.988	0.989	0.987
Hate	0.986	0.986	0.99	0.986	0.987	0.99	0.989	0.993	0.988	0.991	0.987	0.988	0.987
Happiness	0.986	0.987	0.99	0.987	0.988	0.99	0.989	0.993	0.988	0.991	0.988	0.989	0.987
Boredom	0.985	0.986	0.99	0.986	0.987	0.989	0.988	0.993	0.987	0.99	0.987	0.988	0.986
Relief	0.985	0.986	0.989	0.986	0.987	0.989	0.988	0.993	0.987	0.99	0.986	0.988	0.986
Anger	0.987	0.988	0.991	0.987	0.988	0.991	0.99	0.994	0.989	0.992	0.988	0.989	0.988

TABLE XVI: GPT-2 Embeddings Based Similarities

one emotion can focus on the use of adjectives or adverbs for information analysis. Additionally, in a semantic context these adjectives may help in showing strong emotions while adverbs modify sentiment's intensity.

If the percentage of noun tags is large it may give an idea of what the text is about Table III, in this case it may discuss relevant topics, people, places and other noun categories. Similarly the use of two different taggers is to learn the semantic distribution of data between Unigram and Perceptron tagger which consists of 45 tags set.

Table IV shows the vocabulary present in the WordNet database and possible matched and unmatched stats. The possible unmatched percentile ranges between 31% to 37% which shows that roughly 1/3rd of the tweets is not recognized by the lexicon, perhaps that does make sense as tweets are raw, comprising of newly coined words and abbreviations that may not be present in WordNet, however, 63-69% data matched. This does give some unique insight into the type of public data we are dealing with.

In Table V, our goal is to investigate the relationship between words found in the NRC Emotion Lexicon and those present in the Twitter dataset. To achieve this, we calculate the correlation between the vectors representing the Twitter dataset, which includes words occurring in the NRC lexicon, and vice versa. According to the table, all the correlations are positive which shows that they have a positive linear correlation, with 'empty' consisting of the highest correlation. On the other hand, the very low P-values represent that there is negligible sign of randomness. 'Neutral' and 'Worry' with 0 p-value resembles no randomness at all. Ultimately, this table highlights the positive relationship between the words in either datasets.

For Table VI and Table VII, the VADER and sentiStrength analyzers predict the correctly and incorrectly labeling of the Twitter dataset with most categories having more correct matches than incorrect ones. One more important thing to note is that the analyzers disagreed on the sentiments "Empty" and "Surprise". VADER classified "Empty" as negative and "Surprise" as positive whereas SentiStrength classified both "Empty" and "Surprise" as neutral. Both analyzers had similar matching score on other sentiments apart from these ones. SentiStrength had a higher matching score with classifying "Empty" as neutral. Vader had higher matching score with classifying "Surprise" as positive. In summary, it could be inferred that although these analyzers gave us an in-depth idea on the tweets and their sentiments classes but their results cannot be fully trusted, one of the reason being that there would be a lot of sarcasm and weird ways of representing emotions that these analyzers may have a hard time to understand and interprete correctly.

The results shown in Table VIII refer to the Wu & Palmer similarities between all possible categorical pairs. This table shows very important information on Wu & Palmer similarity as well as the WordNet lexicon. The results show a similarity between 'Empty' and 'Relief', 'Enthusiasm' and 'Sadness' or 'Enthusiasm' and 'Happiness', 'Hate' and 'Love' or 'Hate' and 'Anger', 'Boredom' and 'Love', the key detail here is to note that there is a similarity between 'Hate' and 'Love', as well as between 'Enthusiasm' and 'Happiness' which shows that the similarity we are using depends upon the WordNet database. Since, WordNet follows a hierarchical structure, the similarity is scoring based on its relationships such as hyponyms, homonyms, synonyms, and antonyms. As we know 'Love' and 'Hate' are opposite to each other therefore, suggesting that this similarity calculation may not be very efficient for the type of text we are dealing with.

Similar to Table VIII, Table IX also represents the similarities, however, this time the similarities are calculated from Word2Vec vector embeddings and are calculated using Cosine similarities, the key difference here from the previously used Wu & Palmer similarity is that the merging categories being suggested are: 'Hate' and 'Empty', 'Surprise' with 'Enthusiasm', 'Worry' and 'Boredom', 'Neutral' and 'Anger'. This output does make some sense as we can expect 'Hate' to resemble some bit with 'Empty' and similarly, 'Enthusiasm' with 'Surprise' which shows that this similarity calculation technique using Word2Vec is giving meaningful results.

Table X, innovates the idea of calculating similarities, it calculates the similarity between each individual record at word level for every categorical pair as well as on the tweet level, both of those relationships are seen in the table. The diagonal matches like *(Empty, Empty), (Sadness, Sadness),...* represent the word and list based similarities between the classes while the off-diagonal cells represent various comparisons between different sentiment categories.

It can also be seen that the *(Anger, Anger)* category has the highest score among all its pairs.

XI, shows the results of GloVe model, the similarities for each category and their relationships are very high and this could be expected for several reasons, firstly, we are using a model that has been trained on millions and millions of tweets, unlike Word2Vec it considers the entire corpus while learning word vectors. It aims to capture global word relationships. Secondly, since the data we are dealing with are tweets and many of the emotions are similar it is easier for the model to understand the context and score it nicely. In short, high cosine similarities between sentiment categories suggest that there is some commonality in the language used in expressing different sentiments. This could be due to shared vocabulary, semantic relationships, or similar contextual usage.

Table XII, shows the FastText embeddings and the similarity scores for all these embeddings are the highest we have seen, that is most likely the case due to FastText's vast pre-trained data and its understanding of semantics.

For Table XIII, BERT model was used and it gave quite positive results, however, these results could have been better, it also shows that why we have different variations of BERT which are better and faster.

Table XIV, XV, and XVI, these are the state of art approaches that we used to understand and explore different models, and to our surprise the RoBERTa variation of BERT perfromed a lot better than it, similarly the DistilBERT which is supposedly lighter than BERT had a better performance. Finally, the GPT-2 module had almost perfect results, showing the accuracy of the model in learning semantic relationships.

V. Conclusions and perspectives

Completed tasks, used methods, and achieved results have been presented in this paper. Evaluation for the methods has also been presented. Results have been plotted utilizing multiple different figures, histograms and tables and then explained in chapter IV, results discussion and evaluation. Our experiments show that the capability of computer programs in deciphering human emotions out of text is progressing steadily, and can already give useful information although the classification and analysis made by computers is not yet fully accurate. As we can see from the tables VI and VII, these analyzers still make errors in approximately 30% of cases. This percentage will continue to go down in the future and understanding human emotion en masse with computer programs will be a vital part of understanding the general public's opinion and emotional response on world events.

The current state of emotion analyzers is good for analyzing large human labeled datasets but it should only be used as a tool. The results from these methods are mostly correct, which can help you draw correct conclusion from the data you otherwise would not have been able to go through. Languages around the world are constantly evolving and NLP methods are advancing along with the advancement of technology. With the future advances in the field of NLP, these methodologies will catch up to the slow evolution of language and potentially adapt to even faster evolution of slang terms.

VI. Project repository

GitHub repo was used for this project which is as follows: *https://github.com/Lumiin0us/Kaggle_Text_to_Emotion/tree/main*

References

[1] Shelke, N., Chaudhury, S., Chakrabarti, S., Bangare, S. L., Yogapriya, G., & Pandey, P., "An efficient way of text-based emotion analysis from social media using LRA-DNN", in Neuroscience Informatics, vol. II, issue III, 2022

[2] What is WordNet?, https://wordnet.princeton.edu/

[3] Z. Wu and M. Palmer, "Verb Semantic and Lexical Selection," in Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistic, 1994, pp. 133–138

[4] What is Word2Vec, https://code.google.com/archive/p/word2vec/

[5] What is FastText, https://fasttext.cc

[6] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, vol. 31, no. 6, pp. 1532–1543, doi: 10.3115/v1/D14-1162.

[7] What is NRC Emotion Lexicon,https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

[8] Ali Yadollahi, Ameneh Gholipour Shahraki, and Osmar R. Zaiane. 2017. Current State of Text Sentiment Analysis from Opinion to Emotion Mining. ACM Comput. Surv. 50, 2, Article 25 (March 2018), 33 pages. https://doi.org/10.1145/3057270

[9] Hutto, C., and Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. Proceedings of the International AAAI Conference on Web and Social Media, 8(1), 216-225. https://doi.org/10.1609/icwsm.v8i1.14550

[10] What is SentiStrength, http://sentistrength.wlv.ac.uk

[11] Jiawei Han, Micheline Kamber, Jian Pei, 2 - Getting to Know Your Data, Editor(s): Jiawei Han, Micheline Kamber, Jian Pei, In The Morgan Kaufmann Series in Data Management Systems, Data Mining (Third Edition), Morgan Kaufmann, 2012, Pages 39-82, ISBN 9780123814791, https://doi.org/10.1016/B978-0-12-381479-1.00002-2. (https://www.sciencedirect.com/science/article/pii/B9780123814791000022)