COM3029
NATURAL LANGUAGE PROCESSING
INDIVIDUAL REPORT

by

MANUEL BRADICIC
URN: 6574309

GROUP 8

April 2023

Department of Computer Science
University of Surrey
Guildford GU2 7XH

Module Leader: Dr Bogdan Vrusias

Contents

List of Figures

List of Tables

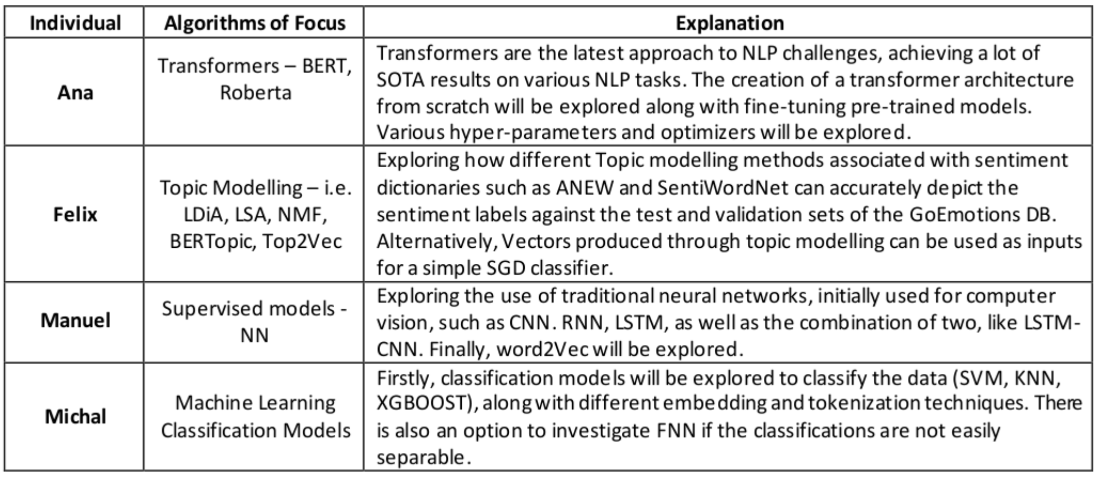
1 Group Declaration - Project Plan

1.1 Labels

The labels that we chose are: Neutral, Anger, Sadness, Admiration, Gratitude, Fear, Surprise, Joy, Confusion, Curiosity, Desire, Disgust, Relief, Remorse. There are two reasons for choosing those labels. Firstly, the plan of the project is to develop a service that will classify product reviews, and these labels give us a good range and balance of emotions. Secondly, we used an Emotion Wheel to combine the remaining labels into these labels. We tried to group them all into parent emotions and tried avoiding any overlaps of emotions.

1.2 Plan for the experiments

The table below shows the NLP method focuses for each member. The necessary data-preprocessing and hyper-parameters will be determined on an individual basis. Each of the members will focus on one of these 4 subtopics – transformers, supervised, unsupervised and ML models.



Individual	Algorithms of Focus	Explanation
Ana	Transformers – BERT, Roberta	Transformers are the latest approach to NLP challenges, achieving a lot of SOTA results on various NLP tasks. The creation of a transformer architecture from scratch will be explored along with fine-tuning pre-trained models. Various hyper-parameters and optimizers will be explored.
Felix	Topic Modelling – i.e. LDiA, LSA, NMF, BERTopic, Top2Vec	Exploring how different Topic modelling methods associated with sentiment dictionaries such as ANEW and SentiWordNet can accurately depict the sentiment labels against the test and validation sets of the GoEmotions DB. Alternatively, Vectors produced through topic modelling can be used as inputs for a simple SGD classifier.
Manuel	Supervised models - NN	Exploring the use of traditional neural networks, initially used for computer vision, such as CNN. RNN, LSTM, as well as the combination of two, like LSTM-CNN. Finally, word2Vec will be explored.
Michal	Machine Learning Classification Models	Firstly, classification models will be explored to classify the data (SVM, KNN, XGBOOST), along with different embedding and tokenization techniques. There is also an option to investigate FNN if the classifications are not easily separable.

1.3 Development environment

- Development Environment: GitHub (code repository); Conda (package management); Jupyter Notebook (code)
- Python Libraries: NLTK, Hugging Face (Transformers), PyTorch, Pandas, Numpy, Sci-kit Learn, Flask-API, H20, Gensim

1.4 Plan changes

Despite the project plan in the table presented in section 1.2 claiming that my focus would involve vanilla RNN, a combination of two, like LSTM-CNN and word2Vec, a modification in my plans settled by the group led me to use pure LSTMs, GRU and GloVe word embeddings instead. The reason for this change is that my colleagues are using similar techniques in their experiments, and we beleived that having some overlap in multiple domains would allow us to make more effective comparisons of our results - for instance, GRU model with GloVe embedding and transformer-based embedding.

2 Data Analysis and Visualisation

2.1 Data Exploration

GoEmotions is a human-annotated dataset, proposed by Demszky et al.[1], of 58,000 Reddit comments extracted from popular English-language subreddits and split into 27 categories. The categories are split across 12 positive, 11 negative, and 4 ambiguous emotion categories, and 1 neutral, adopting it to be applicable in various conversational understanding tasks that involve a subtle differentiation between emotional expressions.

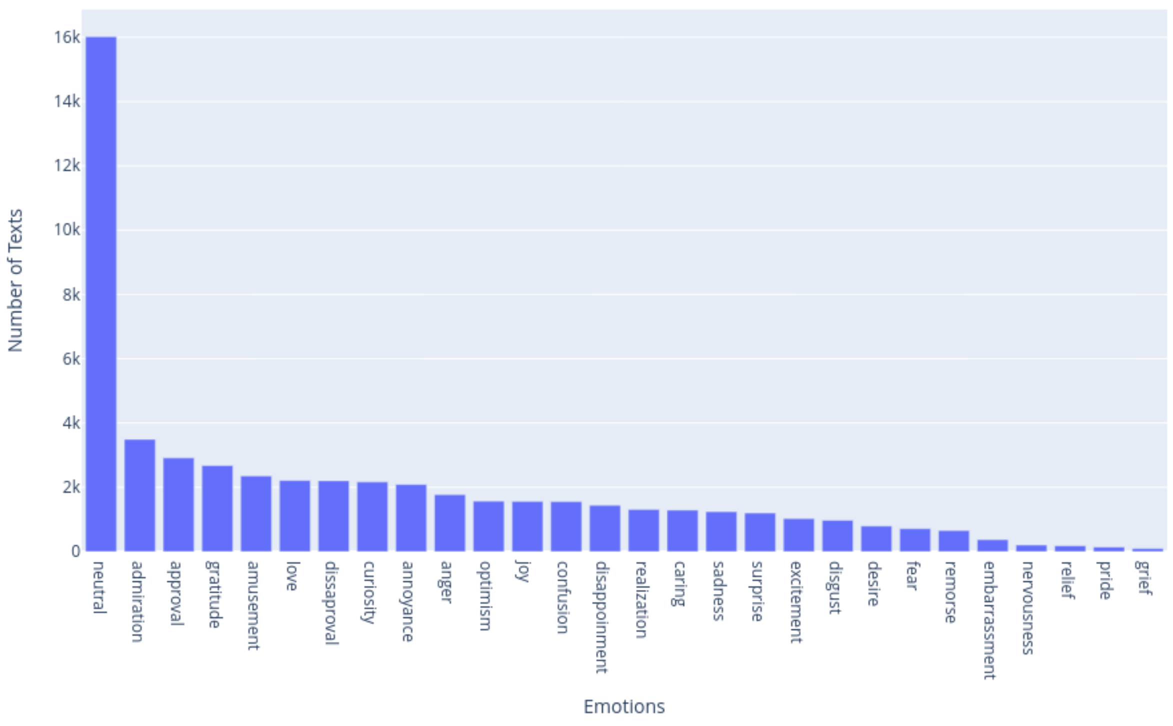


Figure 1: Distribution of labels across the categories

Figure 1 suggests that the number of texts are not evenly distributed across all the classes. This property of a database, when it is not distributed uniformly - also known as an unbalanced dataset, brings some of the challenges that one has to deal with during the training process. For instance, class 'nervousness', as the 4th class with the lowest number of texts, contains only 206 texts, out of the 54137 in total, in other words, it contributes only to 0.38% of the dataset. Having an unbalanced dataset disturbs the training process of most of the models, thus one can expect that most of those low-count labels are going to be predicted as false-negative (FN), however, that is to be investigated.

Another characteristic of GoEmotions is multi-label data. Unlike normal classification tasks where class labels are mutually exclusive, multi-label classification requires predicting multiple mutually non-exclusive labels. Although GoEmotions is a multi-label database, in this work, it will be treated as a multi-class database, which implies that a mechanism for updating a text from multi-labelled to single-labelled has to be developed. Furthermore, as previously mentioned, the database consists of 27 + 'neutral' labels, however, in this work some of the labels will be merged and make up for 13 + 'neutral' labels in total.

During the analysis of the "GoEmotions" dataset, it was found that 134 rows contained duplicate entries with phrases such as "Thank you" or "Happy cake day!". These duplicates were identified based on their exact match with these phrases and were subsequently deleted from the dataset. The removal of these duplicate rows has ensured that the dataset is free from redundant information with the aim of improving the accuracy and reliability of any further analysis or modeling using the dataset.

2.2 From a multi-labelled to a multi-class dataset

Out of the 54,137 rows, or texts that exist in the database, there are 8,817 multi-labelled rows containing 9,549 additional labels. There are several methods that one could use in this scenario:

(A) combining some labels into one

(B) duplicating the records such that each record contains one of the predefined labels

(C) selecting only one of the labels

(D) etc.

Method (A) will be partly achieved during the merging process mentioned above, thus this method will not be used for solving the multi-labelled problem, as it would probably solve only partly the problem, however, it will contribute once merged. Method (B) is one of the methods that should be avoided in this scenario as it provides confusing inputs to the model training process increasing the ambiguity, this method would most likely decrease the accuracy of the model. Method (C) can be developed in many different ways, one can select randomly a label or logic rules for selecting a label could be developed. In this work, the winning label is decided based on the frequency of the labels in the total dataset, in other words, the label with the lowest occurrence in the database is selected as a winning label. This way, text count of the labels which are affected by the property of unbalance is going to be increased.

2.3 Merging the labels

From the initial 27 + 'neutral' labels, this work combines classes into 13 + 'neutral' labels. In order to combine the datasets systematically, preserving their sentiment to some extent, the Emotion Wheel is used. After mapping emotion categories to the Emotion Wheel, a unified emotion can be created by combining and merging the motion categories. This involves identifying overlapping or similar emotion categories. For instance, the label 'annoyance' can be categorised as a subclass of the emotion 'irritable' which is a subclass of the emotion 'anger'. Thus, I combined anger and annoyance. Similarly, 'optimism', 'joy', 'excitement' and 'amusement' can be categorised as a subclass of the emotion 'joy'.

The following pattern was used for merging the labels:

1. Anger –> Anger + Annoyance
2. Sadness –> Grief + Sadness + Disappointment + Disapproval + Embarrassment
3. Admiration –> Acceptance + Approval + Admiration
4. Gratitude –> Gratitude + Love + Pride + Caring
5. Fear –> Fear + Nervousness
6. Surprise –> Surprise + Realization
7. Joy –> Optimism + Joy + Excitement + Amusement
8. Confusion
9. Curiosity
10. Desire
11. Disgust
12. Relief
13. Remorse
14. Neutral

Figure 2 represents the distribution of labels across the dataset after completing the merge.

The property of unbalance remains, however, one can claim that the difference between the classes is lower - compared to the unmerged dataset. For instance, in the previous setup, the fourth smallest label contributed to only 0.36% of the dataset, whereas, the second smallest label in this environment contributes to 1.21% of the dataset.

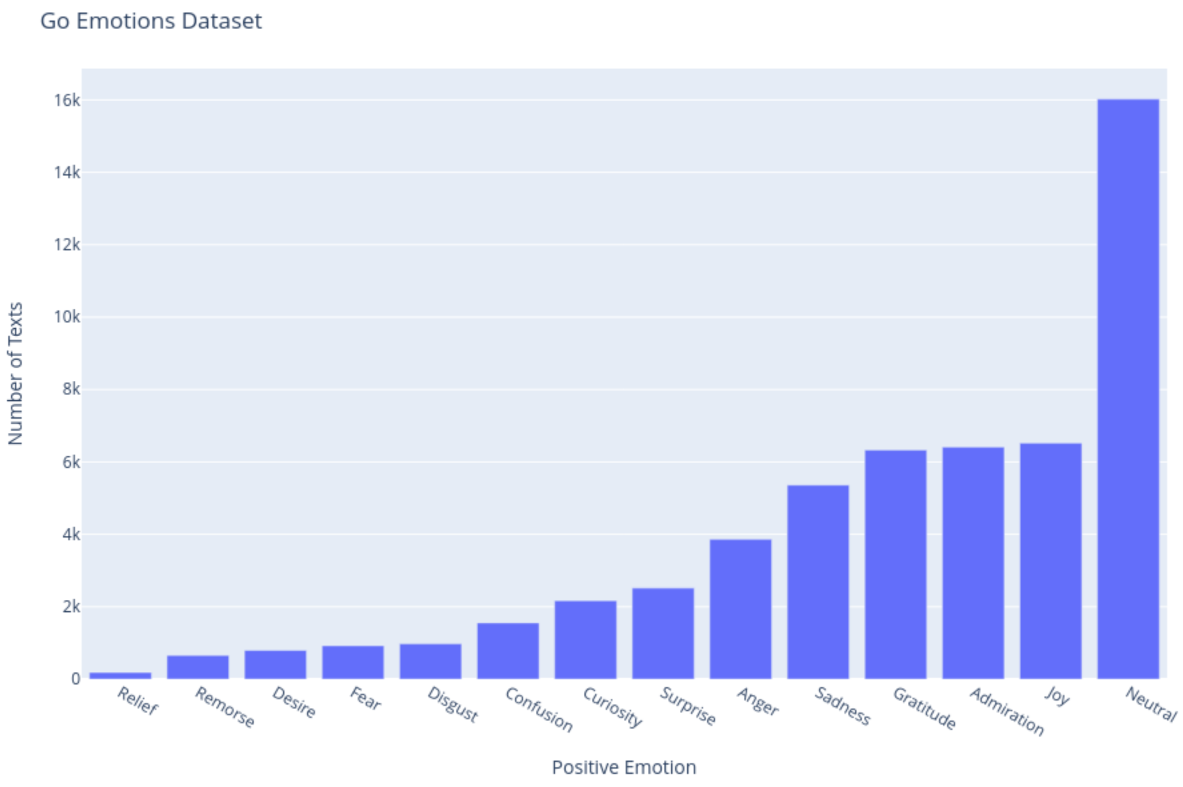


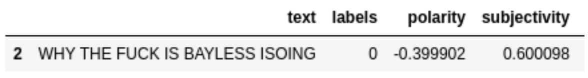
Figure 2: Distribution of labels across the categories after merging the labels into 13+'neutral' classes

2.4 Feature Engineering

Machine Learning algorithms typically require text data to be represented in a numerical form in order to be processed. The research introduced many different methods, however, some of the most common approaches are word2vec, Glove, and Spacy, which map words to real numbers. These word vectors retain the meaning of words by placing similar words in close proximity to each other in a multi-dimensional space. They can be trained on a specific dataset, as well as pre-trained. This work will investigate some of the methods proposed above, as well as pre-trained/trained versions. Another similar method used for representing text numerically is TF-IDF. This section investigates and used basic text features in models as additional features, as well as to visually demonstrate label separability. The text features that will be derived are as follows: Polarity, Subjectivity, Word count, Part-Of-Speech (POS) tags, Uppercase ration, Digits.

2.4.1 Polarity

Polarity or sentiment typically refers to the sentiment or emotional tone conveyed by the text within the range [-1.0, 1.0]. -1 denotes negative sentiment, 0 neutral and +1 positive sentiment. Since this research works with comments, polarity can provide insights into the overall sentiment or emotional sentiment. For calculating one, the TextBlob library was used [2].



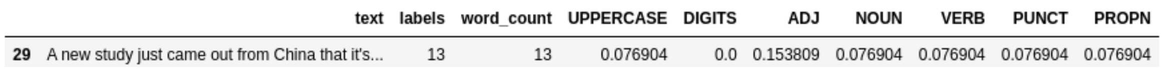
	text	labels	polarity	subjectivity
2	WHY THE FUCK IS BAYLESS ISOING	0	-0.399902	0.600098

Table 1: An example of polarity and subjectivity

For instance, the comment displayed in Table 1 shows that the tone used in the comment was aggressive and negative, due to both uppercase and swear words. Finally, this was confirmed by TextBlob which marked this comment with a polarity of -0.4, in other words as a negative sentiment - which it is!

2.4.2 Subjectivity

Subjectivity refers to the extent to which the text expresses personal opinions, beliefs or emotions. It's expressed by a float in the range [0.0, 1.0] indicating objective and subjective opinion respectively. Similarly, the comment in Table 1 displays a subjectivity of 0.60 which means that TextBlob marked this comment as mostly subjective rather than objective comment, with which I firmly agree.



	text	labels	word_count	UPPERCASE	DIGITS	ADJ	NOUN	VERB	PUNCT	PROPN
29	A new study just came out from China that it's...	13	13	0.076904	0.0	0.153809	0.076904	0.076904	0.076904	0.076904

Table 2: Part-Of-Speech tags

2.4.3 Part-Of-Speech tagging

The ratio of distinct Part-Of-Speech (POS) tags to the total number of words in a given comment:

The following tags are used:

- PROPN - proper noun. Examples: Mike, Jamie, BBC, London
- PUNCT - punctuation. Examples: ., ;, :, (,), ?, !
- NOUN - noun. Examples: baby, tree, bread
- ADJ - adjective. Examples: large, dull, red, first
- VERB - verb. Examples: grow, drink, think
- UPPERCASE - the ratio of the numbers of uppercase words to the word count in given comment.
- DIGITS - the ratio of the number of digits to the word count in given review

2.5 Analysis 1 - Polarity Distribution

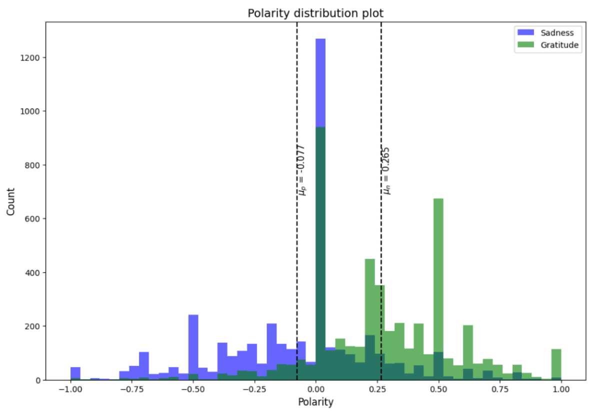


Figure 3: Polarity of sadness and gratitude

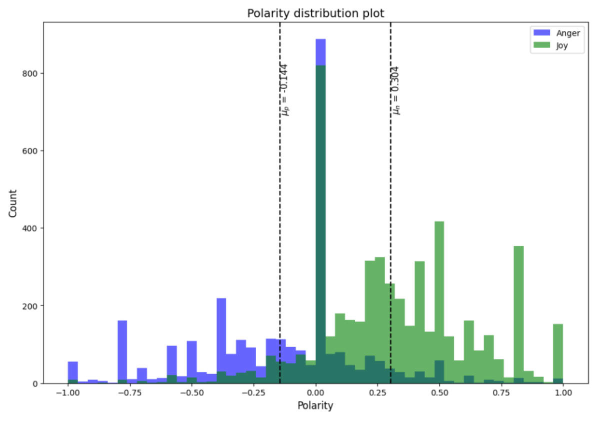


Figure 4: Polarity of anger and joy

From Figure 3, we can draw the following conclusion: reviews which are expressing sadness (coloured in blue) have negative polarity, a mean of -0.077, which means that the TextBlob polarity function assigned almost neutral sentiment (near zero) to the most of the sad reviews. On the other hand, comments expressing joy have a mean value of 0.265. It is visible from the graph that the majority of the outliers dominate the left and right side of the mean for sadness and joy emotions respectively. Similarly, Figure 4 displays the polarity distribution between other two contrastive emotions: anger and joy. These results suggest that using polarity as a classification between positive and negative comments would yield somewhat good accuracy.

2.6 Analysis 2 - Subjectivity Distribution

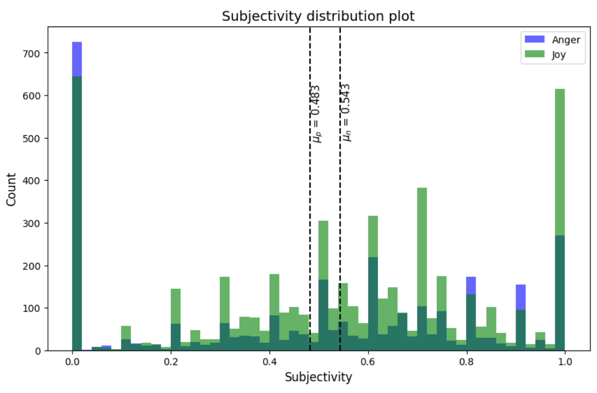


Figure 5: Subjectivity of anger and joy

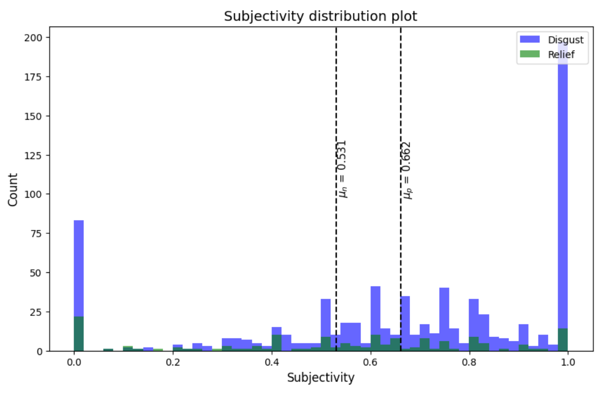


Figure 6: Subjectivity of disgust and relief

As opposed to the polarity distribution, on the subjective score plot, we can observe that the distribution of comments expressing anger and joy greatly overlaps (Figure 5). Subjectivity is roughly evenly spread across the range, which suggests that no matter what comment was given by the author, the comments are almost equally objective/subjective. Similarly, as it was depicted in Figure 6, emotions of disgust and relief overlap with disgust having a slightly higher tendency towards subjective. In conclusion, the subjectivity score feature will not impact the model's ability to predict classes.

2.7 Analysis 3 - Word Count Distribution

In the following analysis, labels gratitude and joy are compared. From Figure 7(a), one can draw the conclusion that the word count for both gratitude and joy labels are very similar to each other which yields a very similar distribution for both classes. In conclusion, the frequency distribution for both categories has a higher frequency of occurrence towards the higher end of the scale, indicating that the median value is located on the left of the average value, in other words, it shows a skewed right distribution. Finally, comments expressing joy tend to have a

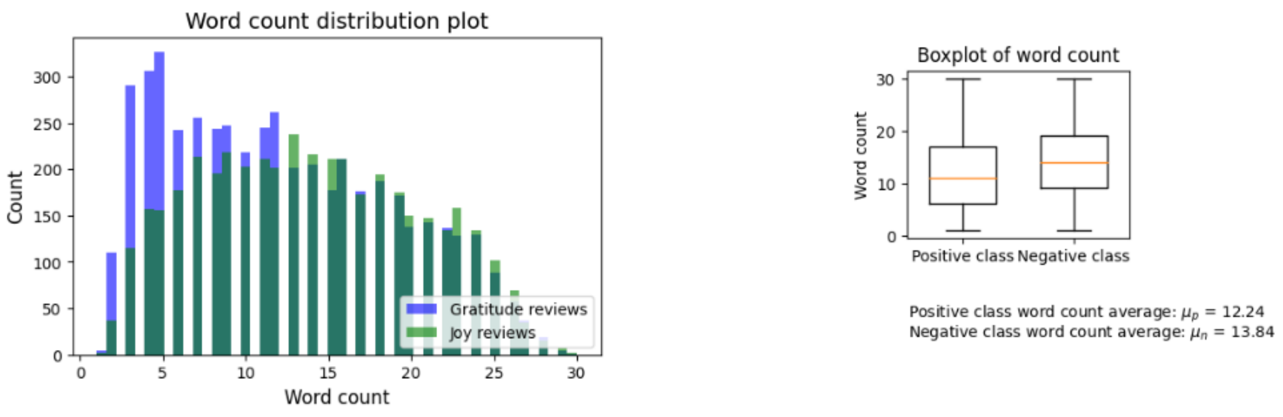


Figure 7: Word count of text labelled as emotions joy and gratitude were analysed; (a) represents word count distribution (b) boxplot of word count

slightly wider spread and outliers, in the sense of word counts, while in comments expressing gratitude this occurrence is fewer. As visible in Figure 7(b), it provides a visual summary of the central tendency spread, and skewness of the word count values for labels gratitude and Joy.

2.8 Analysis 4 - t-Distributed Stochastic Neighbor Embedding

I utilized t-SNE (t-Distributed Stochastic Neighbor Embedding) to represent my data in a two-dimensional space.T his approach is based on G. Hinton and S. T. Roweis [3]. SNE works by converting the high-dimensional Euclidean distances into conditional probabilities which represent similarities [4]. This technique, although computationally intensive and advanced, allows for a visual representation of the data. However, due to the high dimensionality reduction and having categorical data of 14 labels, it makes it challenging to identify any discernible patterns (Figure **??**. Therefore, we will focus on two specific labels, namely sadness and joy, as I have done in previous examples.

In contrast to Figure 8 which represents the two-dimensional space of the entire GoEmotion dataset, the plot displayed in Figure 9 showcases the successful reduction of dimensionality using the t-SNE algorithm for class 0 and 3, anger and gratitude, respectively. The embedded representation of the transformed features demonstrates a moderate level of separability between categories - notably, the bottom left and bottom right parts of the plot are dominated by the 'anger' label, while the 'gratitude' label is scattered across the center of the plot.

In conclusion, the class separability in the current example is moderate, allowing for the possibility of fitting a decision boundary that can reasonably separate two classes. However, considering the presence of 14 classes and the limitation of a low-dimensional space, attempting to draw a decision boundary would not yield meaningful results

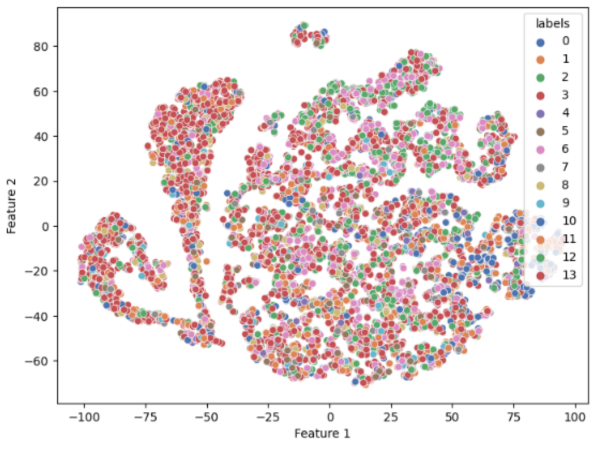


Figure 8: Two-dimensional representation of the dataset using t-SNE

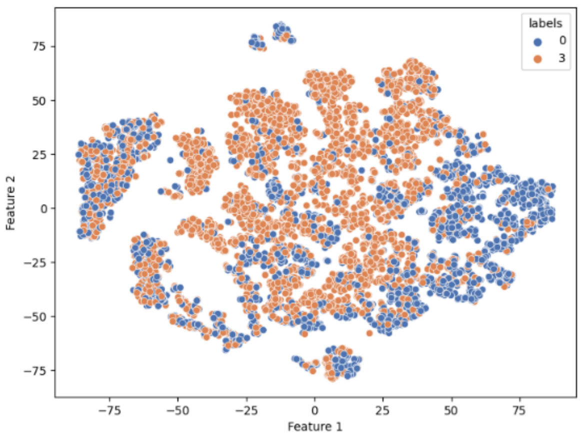


Figure 9: Two-dimensional representation of emotions anger and gratitude using t-SNE

2.9 Analysis 5 - Confusion matrix

The confusion matrix visually provides a visual representation of the correlation coefficients between the features. A coefficient close to 1 indicates a strong positive correlation, while a value of 0 suggests no relationship between variables, and a negative value indicates a negative correlation.

Upon examining the matrix (Figure 11, it is apparent that there is no direct correlation between ADJ/NOUN/VERB/PUNCT/PROPN and word count/polarity/subjectivity. However, it is evident that VERB/NOUN/ADJ are correlated, indicating that changes in these building blocks(i.e., adjectives, verbs, or nouns) may be correlated with changes in the second feature, either in an increase or decrease manner.

2.10 Analysis 6 - Top 20 words

The list of the top 20 words provides us with some insights into the dataset, which appears to involve emotions, opinions and interactions. For instance, words such as ’love’, ’like’, ’think, ’good’, ’say’, and ’thanks’ suggest that the database may contain comments expressing emotions, opinions and sentiments. Whereas, 'go', 'make', 'see', and 'look' may indicate some actions that might be correlated with emotions or people's opinions. Finally, the word 'name' occurs as the most often word, however, performing some data analysis and considering the nature of the dataset, I believe the 'name' tag associates the person to who the author is referring - also known as tagging someone's name.

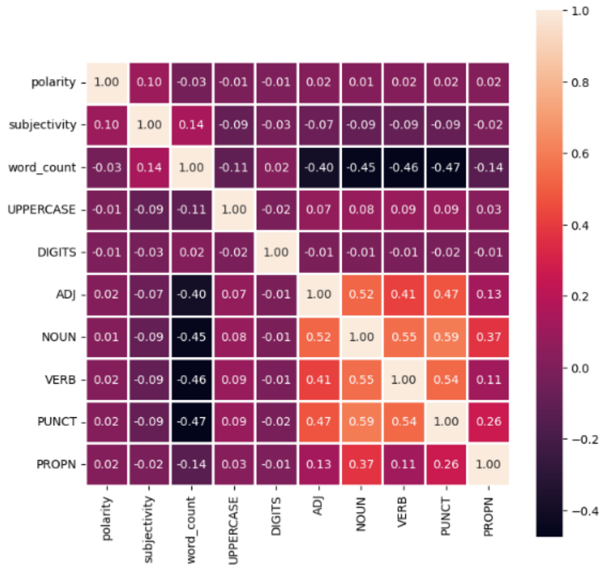


Figure 10: Word count of text labelled as emotions joy and gratitude were analysed; (a) represents word count distribution (b) boxplot of word count

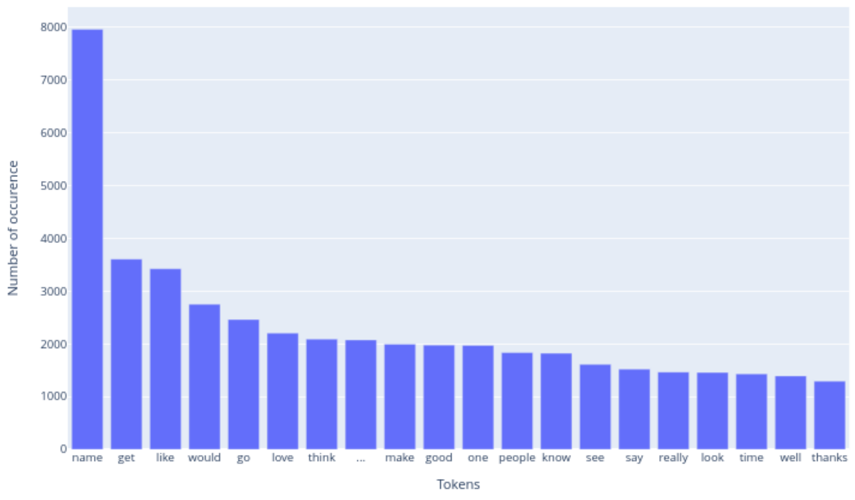


Figure 11: Top 20 words that occurred in the dataset after data pre-processing, Section 3, Experiment I

3 Experiment I - Data Preprocessing Techniques

Feature extraction and pre-processing play a critical role in text classification tasks [5]. In this section the following classical methods were explored and applied to the dataset:

- Expanding Contractions
- Foreign Language Detection
- Grammar detection
- Tokenization Lowercase transform
- Removing Punctuations
- Encoding tokens

	text	labels	id	text_cont_str	tokenized	lower	no_punc	removed_stopwords	pos_tagging	wordnet_pos	lemma	tokenized_padded
1	Now if he does off himself, everyone will thin...	[13]	ed00q6i	Now if he does off himself, everyone will thin...	[Now, if, he, does, off, himself, ,, everyone,...	[now, if, he, does, off, himself, ,, everyone,...	[now, if, he, does, off, himself, everyone, wi...	[everyone, think, hes, laugh, screwing, people...	[(everyone, NN), (think, NN), (hes, NNS), (lau...	[(everyone, n), (think, n), (hes, n), (laugh, ...	[everyone, think, he, laugh, screw, people, in...	[[108, 7, 1113, 210, 996, 13, 322, 60, 401, 0,...

Table 3: An example of data-preprocessing

3.1 Expanding Contractions

Contractions are shortened versions of words or phrases that are commonly used in informal speech and writing, such as "can't" for "cannot" or "didn't" for "did not". Expanding contractions involves replacing these contracted forms with their full form.

Most importantly, expanding contractions ensures consistency in text representations, which can benefit many ML tasks performing NLP by ensuring a cleaner and more standardized input for analysis. For example:

My favourite food is anything I didn't have to cook myself

The same example after being processed:

My favourite food is anything I did not have to cook myself

3.2 Foreign Language Detection

Although the dataset was manually labelled, I still looked into foreign language detection as a part of the pre-processing technique. Having several languages in the database would increase the ambiguity of the dataset which would decrease Machine Learning models' performance on the dataset. Spacy language detection library was used to develop a method that iterates over the dataset and flags foreign languages. However, due to the nature of the dataset - scraped from Reddit, it contains an immense amount of slang words and other speech symbols, such as emoticons, which makes Spacy language model confused. Out of the 20 flagged texts, i.e. texts which were predicted as a foreign language (THRESHOLD <0.7), 20/20 predictions were false positives (FP), which means that the detector misclassified 100% of the samples which were manually investigated. For instance, the following texts were identified as potentially being written in a foreign language:

1. *just noticed, lol. damn pervert foreigners*; score (0.5714)
2. *Dogs, unconditional love* (0.5714)
3. *I did not realise*; score (0.2857)

While the concept of language classification for text is valid, I am afraid that I do not possess the necessary resources to perform computationally intensive language detections with a training process on the dataset. As a result, this feature will not be utilized in the analysis.

3.3 Grammar Detection

Grammar detection is another important step of natural language processing, as it plays a significant role in ensuring the accuracy and reliability of NLP tasks. Slang and abbreviations are an example of grammatically incorrect words which can be misused by the model's embeddings and misclassify the words. Moreover, typos are commonly present in texts and comments, especially on social media such as Reddit. Grammatically correct sentences that are easier to understand and interpret by the models, many NLP techniques rely on the correct grammatical structure of a sentence because it allows embeddings to correctly map the words in multi-dimensional space. In this work, GingerIt library was used to deal with such examples. It is worth mentioning that computationally it is a very expensive operation taking severals hours to iterate through all of the examples.

3.4 Tokenization

Tokenisation is the process of converting a sequence of characters - sentences, into a series of tokens, where each token represents a meaningful element.

My favourite food is anything I did not have to cook myself

In this case, the tokens are as follows:

{'My', 'favourite', 'food', 'is', 'anything', 'I', 'did', 'not', 'have', 'to', 'cook', 'myself' , '.' }

3.5 Lower case

Reddit comments have a wide range of capitalization. Writing in lowercase or upercase might bring some meaning or suggest what was the message behind that review. Usually writing fully uppercase could mean that a person is expressing his anger, or 'shouting' as we usually think of it. However, mixing uppercase and lowercase is another feature which increases the number of vectors in the latent space, but not necessarily an increase in the dimensionality of the problem, thus this work will focus on the lowercase sentences, however, the data frame will also contain tokenized data without lowering all the letters and allow further testings in the experiments. However, this technique brings another problem such as the interpretation of certain words, such as 'US' (United States of America), and the word 'us' as a preposition.

3.6 Removing Punctuations

One could argue that punctuations bring semantic meaning to the sentence, similarly, one can express their feelings through punctuation. For instance, confusion is one of the labels used in the dataset which is very often expressed using question marks. In this work, similar to lowercase, we will reduce the number of vectors in space by removing the punctuations, however, the column containing the punctuations will remain in the data frame for the testing purpose

3.7 Stop words

Text in general includes many words which do not bring any significant importance used in a classification algorithm, such as 'a', 'an', 'are', 'is', 'by', 'then', etc.. The most common approach is to remove these words from the text.

{'My', 'favourite', 'food', 'is', 'anything', 'I', 'did', 'not', 'have', 'to', 'cook', 'myself' }

After the stop word processing:

{'favourite', 'food', 'anything', 'cook' }

3.8 Positional tagging

Positional tagging, also known as part-of-speech (POS) tagging is a process during which each word/token in a sequence is assigned a grammatical tag based on its syntactic properties. Models trained with such data are provided with information about the grammatical structure of a text. Some of those tags are noun, verb, adjective adverb, etc. (See Section 2.4.3 for more)

An example of tokenized sequence with POS tags:

{('favourite', 'RB'), ('food', 'NN'), ('anything', 'NN'), ('cook', 'NN')}

3.9 Lemmatization

Lemmatization involved reducing words to their base or root form, also known as a lemma. This process reduced the variations of words to a common form, which means that all the words and variations or a word, after being processed by an embedding will be placed at the same place in the multi-dimensional vector space. Having several different variations increases the ambiguity of data. I.e., it helps to reduce the size of the vocabulary. See Table 3 column 'lemma' which represents the results of lemmatization for the given example.

3.10 Generating vocabulary

By generating a vocabulary one is obtaining the set of unique words used in the collection of data. After performing all of the preprocessing steps in Section 3.1-3.9, the lemmatized data has been fed into the algorithm and the vocabulary object has been created. The vocabulary serves as the basis for constructing the set of features that are used to represent the data. Each word in the vocabulary can be considered as an id of each word, or as a unique feature used for representing the text in numerical form.

3.11 Disscusion

In order to train models, both LSTMs, which are investigated in Section , and GTP models - current SOTA models, require tokenization. However, the question is how much data processing is needed for models to perform well and at the same time to reduce the ambiguity of the problem.

Data preparation conducted in Sections 3.1-3.9 are classical data preprocessing steps used for working with classical NNs, such as RNNs, LSTMs or convolutional networks in language processing. One can argue that some of these steps reduce the semantic and contextual meaning of words, with which I purely agree. For instance, text written in capital letters often indicates a harsh and aggressive tone, which could be beneficial for a classifier. Similarly, punctuation brings another semantic meaning to the sentence, someone could express their 'confusion' by using question marks. Furthermore, an exclamation mark could indicate excitement or anger. Finally, three dots ('...'), also known as an ellipsis, can be used to express confusion or irony in the comment. The question that this usually rises is how much data can we keep as 'raw' and hope that models extract meaningful information.

RNNs were introduced and used for solving sequential data, however, they suffer from the problem of vanishing gradients, which means that the model creates a bias forward the most recent word in the sequence, and when the gradient becomes too small the parameter updates become insignificant. LSTMs were proposed with that idea in mind, and they do perform better than RNNs, as they've got different kinds of activation functions, known as LSTM cells. However, they still suffer from this problem and over long texts. GPT, on the other side, is a type of transformer-based language model that used a self-attention mechanism, which was proposed in the paper 'Attention is all I need' [6], to process the input data. The biggest advantage of the transformer is that at each step of processing, one can have direct access to self-attention. In comparison to LSTMs/RNNs, transformers process sentences as a whole, rather than treating them token by token, due to the property of parallelisation, which makes them run fast.

Saying this, one can conclude why data preprocessing is essential, bringing down words such as 'running', 'runs, 'run' to a single token, it reduces the number of vectors in the latent space, and at the same time it preserves the core representation of the word. Equally important, removing stopwords (Section [?]) reduces the length of the problem, thus helping models which suffer from vanishing gradient to perform better.

In conclusion, it is necessary to find the balance between preserving the raw data and completing data preprocessing. Different models and different stages of data pre-processing have to be tested in order to conclude how much pre-processing is needed.

4 Experiment II - LSTMs

This section investigated LSTMs and their performance. Various techniques are explored and experimented. In order to feed the data into models, one has to convert textual representation into numerical representation, also known as text encoding. After the data is fed into the model, and processed by the embedding layer each token gets a 'place' in the multidimensional place. Word embeddings are vector representations that capture the semantic and contextual meaning of words. They are usually generated during the training process from scratch on the specific corpus of data, however, they can also be used as pre-trained values. Word2vec, and GloVe are some of the libraries which contain word embeddings trained on a large corpus of data. Both LSTMs with basic embeddings and pre-trained embeddings are investigated.

4.1 Text Encoding

In order to complete the numerical representation of textual data, one has to create a vocabulary and perform encoding. The vocabulary used in this work used *torchtext.vocab* library. The maximum number of words is set to 25,000 with a minimum frequency of 3. One has to set the minimal frequency because, in general, models benefit from reducing the size of vocabulary - it is a cost we accept to take, we eliminate some words which have negligible occurrence in order to have a lower number of vectors in latent space.

For instance:

{'favourite, 'food', 'anything', 'cook'}

Becomes:

{'886', '334', '115', '1466'}

4.2 Analysis 1 - Hidden dim., Embedding dim., Number of Layers

In this experiment, basic LSTMs with fixed-length inputs are investigated. Each network has an embedding layer of dimensions D, and vocabulary size V, which is the size of the initial vocabulary, same for all the LSTMs - 8808. It has an LSTM layer of hidden dimensions H and a number of layers N. All models contain a dropout rate of 0.2 as the last layer. Finally, learning rate lr and epochs e.

In this analysis, my primary goal was to understand how changing parameters affect the learning characteristics of the models. Each model involved only one change compared to model A. See Table 4 for the parameters and Figure 17 for the results.

For model (A) simple parameters were taken as a benchmark, and the highest validation accuracy and loss achieved are 53% and 1.621. Model (B) on the other hand has an increased number of layers to 3. The figure shows the change in the training process, it is visible that the loss started decreasing much later, just before the 10th epoch. My assumption is that LSTM with an increased number of layers can capture more abstract and higher-level representations of data, however, due to the increase of parameters, the model might require a lower learning rate in order to generate valuable results. This model achieved the highest accuracy. Model C with an increased embedding size takes slightly shorter to start decreasing the loss. Similarly to model (B), increased dimensionality (an increase in parameters) requires more computational power, resulting in increased computational time. Furthermore, networks with many layers may suffer from vanishing gradient, which can make training slower and more challenging. Model D yields the worst validation loss, which is not surprising. Increased hidden dimension allows one to learn complex patterns from data, resulting in improved performance on certain tasks, however, larger models come with trade-offs, such as computational complexity and memory requirements. In conclusion, having higher hidden dimensionality doesn't necessarily increase the performance of the model - after a certain point, the accuracy starts decreasing as the dimensionality of the problem becomes exponentially large, and finding the minimum becomes immensely difficult - such as proved in this analysis.

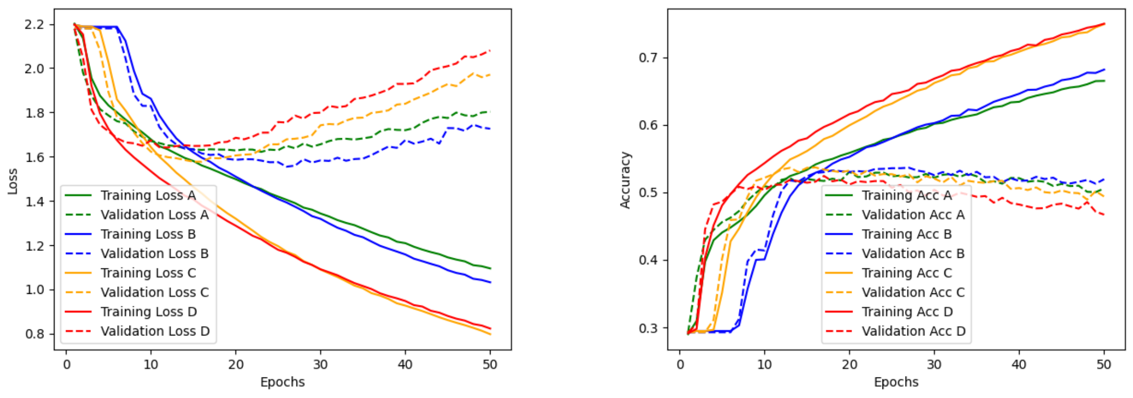


Figure 12: Models A, B, C, D; Training and validation (a) loss and (b) accuracy

	A	**B**	**C**	**D**
Epochs (e)	50	50	50	50
Learning rate (lr)	0.001	0.001	0.001	0.001
Vocabulary size	8808	8808	8808	8808
Hidden dim (H)	50	50	50	128
Embedding dim (D)	50	50	128	50
Number of layers (N)	1	3	1	1
Validation Acc/Loss	53.0/1.621	53.6/1.554	53.7/1.577	52.4/1.641
Test Acc/Loss	52.5/1.648	53.6/1.569	53.3/1.579	52.3/1.65

Table 4: Model's performance depending on the parameters

4.3 Analysis 2 - Hidden Dimension and One-cycle cosine learning rate

Analysis conducted in this example focuses on the hidden dimension of the LSTM model as well as the learning rate. Having a static learning rate, such as in Analysis 1, limits the performance of the models. Research suggests using techniques which decrease the learning rate over the iterations. In this analysis, the one-cycle cosine learning rate was used and models E,F,G,H were tested. One-cycle cosine learning rate is a technique used to optimise the learning rate during the training process. It involves dynamically adjusting the learning rate of a neural network over a number of epochs. The models' parameters are all equal, apart from the hidden dimensions, which are *64, 96, 128, 256*, respectively.

It is visible from Figure 13 that each of the models behaves differently. Looking vertically, from the top to the bottom, between epochs 30 and 40, one can observe that the smallest the dimension size is (green vs red, 64 vs 256), the slower the model converges the minimum. On the other side, it is visible from model H (red) that soon after reaching the minimum, the model starts performing worse - it starts overfitting itself. This is the situation in which the dropout rate plays a role, thus, I will try to increase it. Overall, model performance has increased using this property of one-cycle cosine learning rate (Compare loss in Figure 13 and Figure 12).

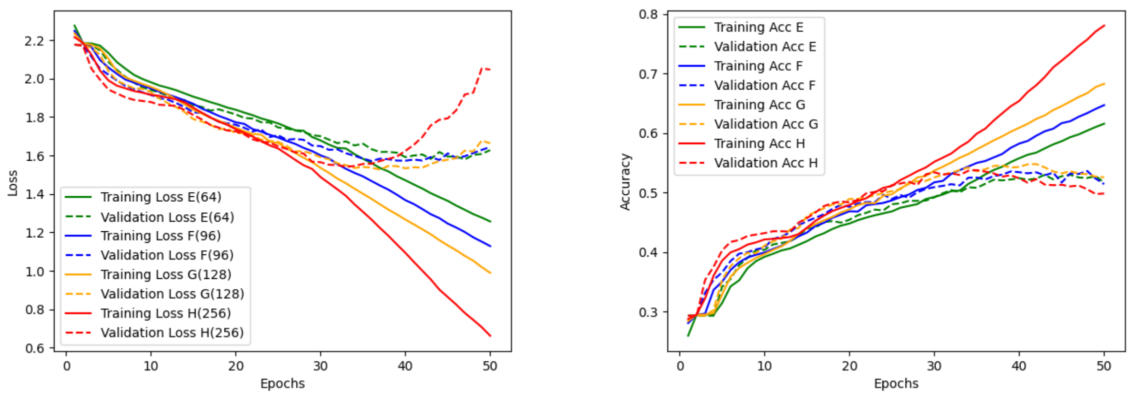


Figure 13: Models E, F, G, H; Training and validation (a) loss and (b) accuracy

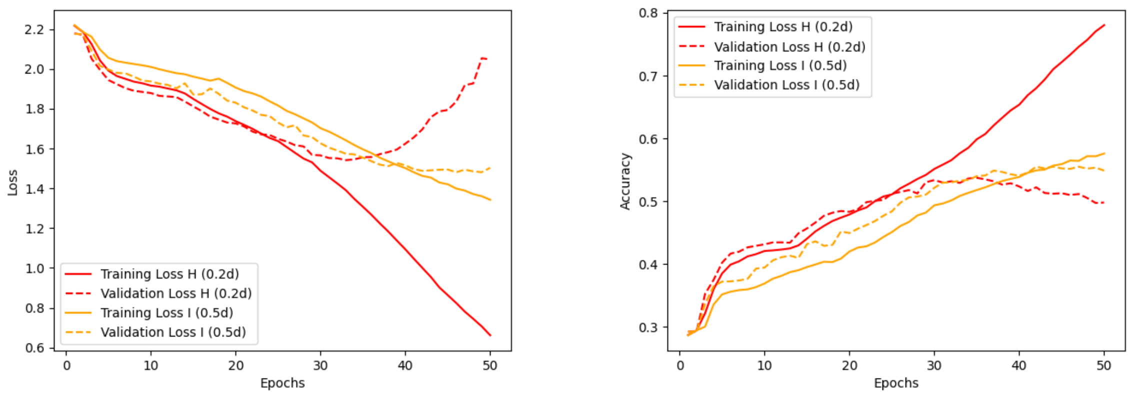


Figure 14: Models I, J; Training and validation (a) loss and (b) accuracy with specified dropout

4.4 Analysis 3 - Dropout rate

Figure 13 represents models E, F, G and H, their training and validation accuracy through the training process. It is visible from the figure (b) that in all 4 cases, training accuracy keeps growing while at the same time model starts performing worse on the unseen dataset - validation. This characteristic is one of the properties of overtraining and can be dealt with in many different ways.

The technique used in this analysis is a dropout rate - it is a very common way of making a network flexible and less drawn to overfitting. Figure 14 represents models H and I. Model H remains with the same parameters that were introduced in Section 4.3, whereas, the only change introduced in the model I compared to model H is the dropout rate - it was increased from 0.2 to 0.5

It is visible from the graph that the behaviour of the model has changed. Model H which has a low dropout rate starts overfitting very soon, whereas the other hand model I which has a dropout of 0.5 learns slower, however, maintains learning and doesn't overfit itself in the given epochs. Furthermore, the same model yielded the highest accuracy measures up to this point, of a total 56.2% and a validation loss of 1.481. On the other hand, the highest accuracy achieved by model H is 54.6%. The model I performed 55.5% on the test dataset.

4.5 Analysis 4 - Embedding dimensions

The following section looks into variations of embedding dimensions. The models used in this example are built upon model I from the previous analysis. In this scenario, we tested embedding dimensions of 50, 100 and 150.

Figure 15 provides us with training and validation loss and accuracy of models with different embedding sizes. Model K achieved maximum accuracy and minimum loos of 56.0% and 1.491, which are the best results obtained up to this point. This experiment proves that increasing the embedding size has a direct correlation to the performance of the model and indicates that model was able to distribute the vectors of tokens better in the bigger latent space yielding higher accuracy. Finally, this analysis proves that increasing embedding dimensions doesn't mean a necessary increase in performance (150 vs 100). Finally, LSTM with variable length input (instead of fixed padding) were tested, however, they haven't yielded any better performance, thus the example won't be explored in detail.

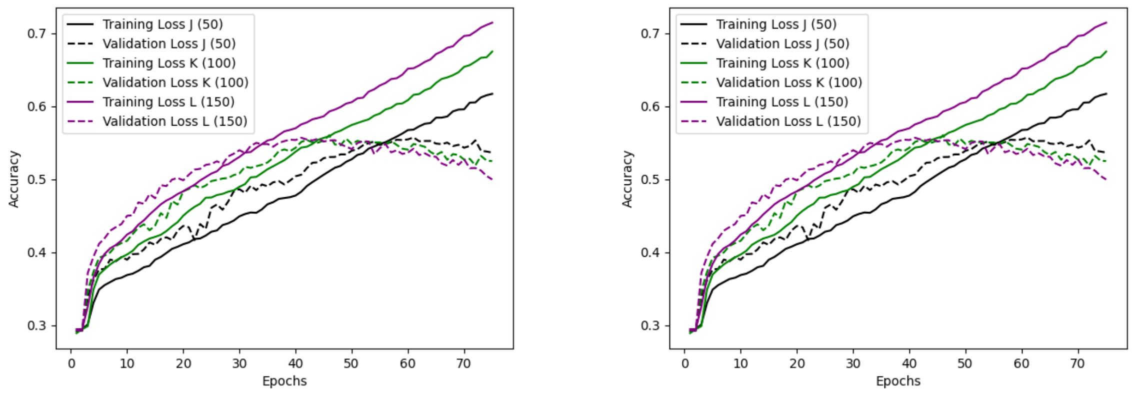


Figure 15: Models J,K,L; Training and validation (a) loss and (b) accuracy with specified embedding size

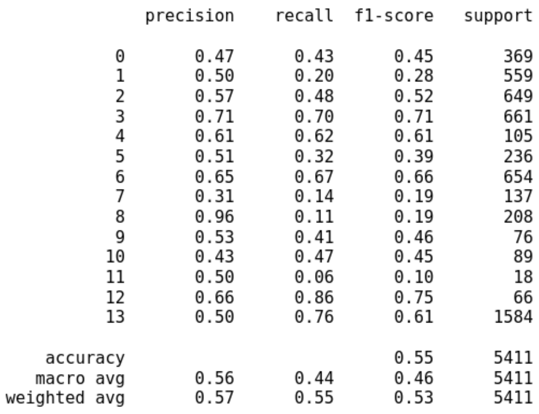
4.6 Analysis 5 - Deeper Study of the Model K

To better understand model K's performance, in this analysis, various performance measurements of the model are provided. Just to remind ourselves, the model has: embedding dimensions 100, hidden dimensions of 256, a number of layers of 3, and a dropout rate of 0.5. In terms of the training process, it was run over 75 epochs, the learning rate was adjusted using a one-cycle cosine learning rate, and the model with the highest validation accuracy was used.

Figure 16 provides a classification matrix and a confusion matrix. What we can conclude from the provided report and the figure is how the model performs per class. For instance, label 12 ('relief') performed significantly well. Its precision score is 0.60 and its recall score 0.86. In terms of precision, it tells us that only 0.60 of the total predicted (TP+FP) is correctly marked. On the other hand, the score of 0.86 for the recall tells us that out of all labels, 86% of them were flagged as label 12. On the other hand, label 8 has a precision of 0.96 and a recall of 0.11. One might think that this is a good score, however, it tells us that label 8 was very often flagged as other labels, however, one positive thing is that it was very rarely confused by the other labels, thus the precision is 0.96.

One might think that this is a good score, however, it just tells us that label 8 was rarely ever flagged. However, a positive thing is that when it was flagged, it was flagged correctly - which is proved by the precision. In terms of the F1 score, label 12 performed the best, while labels 7 and 8 performed the worst.

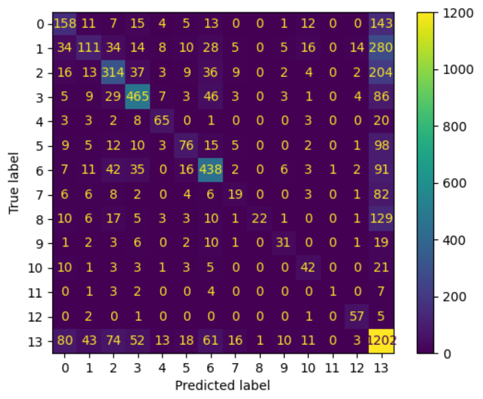
In conclusion, the overall performance of the models investigated in this section is not terribly bad, however, I was personally expecting higher accuracies. I believe embeddings play a big role towards higher accuracies, in the examples above all the embeddings were trained from scratch and purely using our dataset - which, although it consists of almost 54,000 examples, it is still a small database, or rather saying, not big enough to be able to place all the tokens on the 'right' spot in the multidimensional latent space. Therefore, pre-trained embeddings will be investigated next.



```
              precision    recall  f1-score   support

           0       0.47      0.43      0.45       369
           1       0.50      0.20      0.28       559
           2       0.57      0.48      0.52       649
           3       0.71      0.70      0.71       661
           4       0.61      0.62      0.61       105
           5       0.51      0.32      0.39       236
           6       0.65      0.67      0.66       654
           7       0.31      0.14      0.19       137
           8       0.96      0.11      0.19       208
           9       0.53      0.41      0.46        76
          10       0.43      0.47      0.45        89
          11       0.50      0.06      0.10        18
          12       0.66      0.86      0.75        66
          13       0.50      0.76      0.61      1584

    accuracy                           0.55      5411
   macro avg       0.56      0.44      0.46      5411
weighted avg       0.57      0.55      0.53      5411
```



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	158	11	7	15	4	5	13	0	0	1	12	0	0	143
1	34	111	34	14	8	10	28	5	0	5	16	0	14	280
2	16	13	314	37	3	9	36	9	0	2	4	0	2	204
3	5	9	29	465	7	3	46	3	0	3	1	0	4	86
4	3	3	2	8	65	0	1	0	0	0	3	0	0	20
5	9	5	12	10	3	76	15	5	0	0	2	0	1	98
6	7	11	42	35	0	16	438	2	0	6	3	1	2	91
7	6	6	8	2	0	4	6	19	0	0	3	0	1	82
8	10	6	17	5	3	3	10	1	22	1	0	0	1	129
9	1	2	3	6	0	2	10	1	0	31	0	0	1	19
10	10	1	3	3	1	3	5	0	0	0	42	0	0	21
11	0	1	3	2	0	0	4	0	0	0	0	1	0	7
12	0	2	0	1	0	0	0	0	0	0	1	0	57	5
13	80	43	74	52	13	18	61	16	1	10	11	0	3	1202

Figure 16: Model K (a) Classification Matrix (b) Confusion Matrix

5 Experiment III - GloVe Vectors

Global Vectors for Word Representation, aka GloVe, is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space [7].

5.1 Understanding GloVe

GloVe is an unsupervised technique which represents words as vectors in a high-dimensional space, also known as latent space. It is a pre-trained word embedding model, trained on a large corpus of text data. In this example, 3 GloVe embedding dimensions are investigated, 50, 100 and 200. They are all pre-trained models and downloaded from the official website (https://nlp.stanford.edu/projects/glove).

To better understand the 50-dimensional latent space of GloVe, I tried to find the 5 most similar words to the word 'fun'. What I got are: *'crazy', 'stuff', 'wonderful', 'imagine', 'terrific'*. Another example if we consider is math operation. For instance, summing up the vectors' of the words 'game' and 'console' give us the vectors of the words 'xbox' and 'playstation'. This shows that the words are positioned in some mathematical and logical order in the latent space.

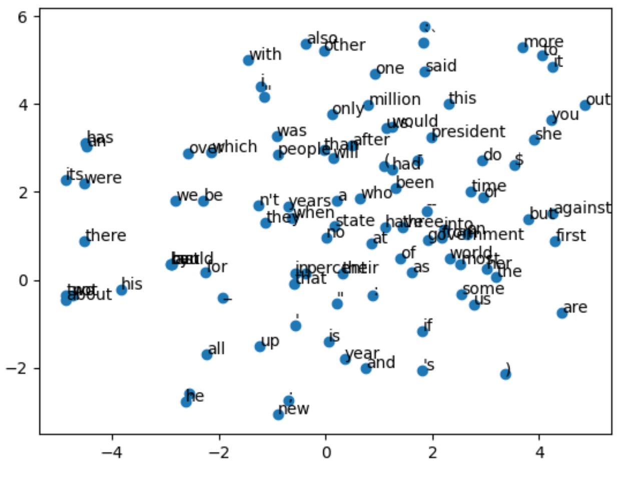


Figure 17: 2D representation of Glove 50-dimensional embedding for first 100 vectors

5.2 Pre-trained vs trained embeddings

In this section, both models trained using GloVe embeddings and those trained from scratch are investigated and compared.

Firstly, let's take GloVe with the embedding dimensions 50 as an example. It is a pre-trained model, trained using a large corpus of data, which includes vectors of around 400,000 tokens. In other words, its matrix size is *400,000 x 50*, which means that for each token, there are 50 vectorial values that describe each token.

All the models were trained using parameters equal to the model K - hidden dimensions of 256, number of LSTM layers of 3, and drop out of 0.5. However, the models were run over 75 epochs, as well as different embedding dimensions were investigated. Model F was run using the embedding size of 50 and it achieved the best validation accuracy/loss of 51.6% and 1.601, respectively. Model G was trained using an embedding dimension of 100, the model's performance improved and it achieved 53.0% and 1.564. Model H was run using an embedding size of 200, the highest obtained validation accuracy and loss are 54.6% and 1.516. Finally, model O was compiled using the embedding of dimensions 300. The model converged at the validation accuracy and loss of 54.8% and 1.485.

The first characteristic of the GloVe embeddings which I noticed is that the higher the embedding dimensions are, the longer it takes for the model to be trained. This is an expected behaviour as the matrix dimensions increase, thus it becomes computationally more expensive. However, longer training does not necessarily mean that the models performance is going to be better. Figure 18 displays the overall models' performance on validation dataset using pre-trained embeddings - models R,N,O,P and compared to the performance of a model trained from scratch - model K. It is visible from the figure that model K, which was trained purely using GoEmotions dataset performed the best.

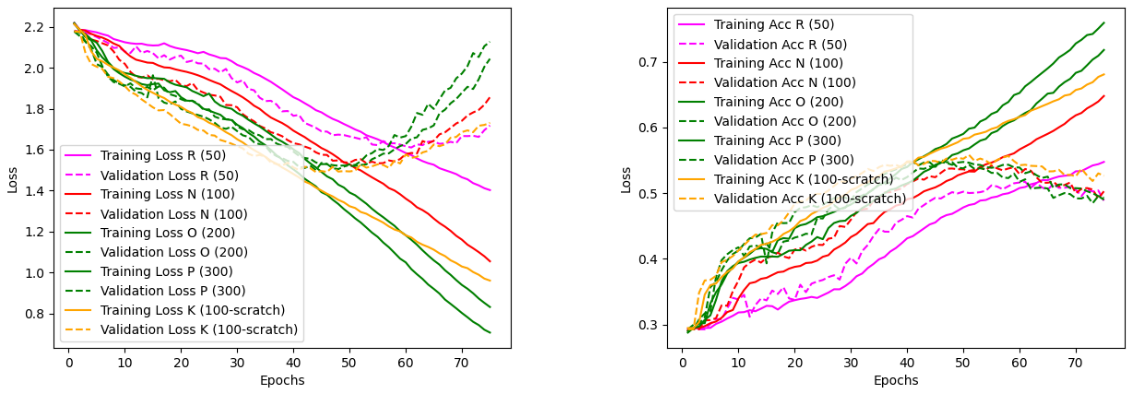


Figure 18: Models J,K,L; Training and validation (a) loss and (b) accuracy with specified GloVe embedding size

6 Experiment IV - Gated Recurrent Unit

Gated recurrent units (GRUs) are a gating mechanism in RNNs, introduced in 2014 by Cho et al. [8]. A GRU is like an LSTM with a forget gate but has fewer parameters than LSTMS, thus is trained much faster compared to LSTMs.

The reason for implementing GRUs is that research suggests that GRUs overall perform better on smaller datasets, while LSTMs perform better on big datasets.

6.1 Training process

To train the GRUs, BatchIterator class was introduced, this class allows us to train models in batch sizes, rather than iterating through data loaders directly. Each batch consisted of 64 rows. In this example, vocabulary is recreated during the training process on each batch, rather than generated for the whole dataset - the minimum word count remained 3. Finally, GloVe embeddings were used.

The first thing noticed during the training process is that it takes marginally shorter to train GRUs compared to LSTMS. GRUs with 100, 200 and 300 embedding sizes were tested.

6.2 biGRU GloVe 300

GRUs using GloVe 300,200 and 100 were investigated, and GloVe 300 yielded the best results. The parameters used are similar compared to the previous ones used in model K, with hidden dimensions of 128, number of layer 3, dropout 0.5 and learning rate of 0.001. Finally, bidirectional training was used too, which means that, unlike the standard LSTMS, the input flows in both directions. Moreover, early stopping was introduced, this technique prevents model the model from running if the overall performance is getting worse by every epoch, it stops the model running after X epochs.

6.3 Performance analysis

The highest accuracy achieved using this model is 53.6%. It is debatable why most of the models perform similarly, however, it is further discussed in section 8 Evaluation. If we compare the model's performance with model K, using the confusion matrices displayed in Figure 19 similar pattern can be noticed. For instance, classes 3, 6 and 13 perform the best overall, with a similar probability (colour intensity in the matrix). Moreover, it is visible that a great portion of comments are classified as the label 13, in both types of models, which however is not correct. This suggests that label 13 - neutral, brings some ambiguity to the model, thus it classifies labels as 13 instead of others. This could be because of the data distribution explained in data visualisation.

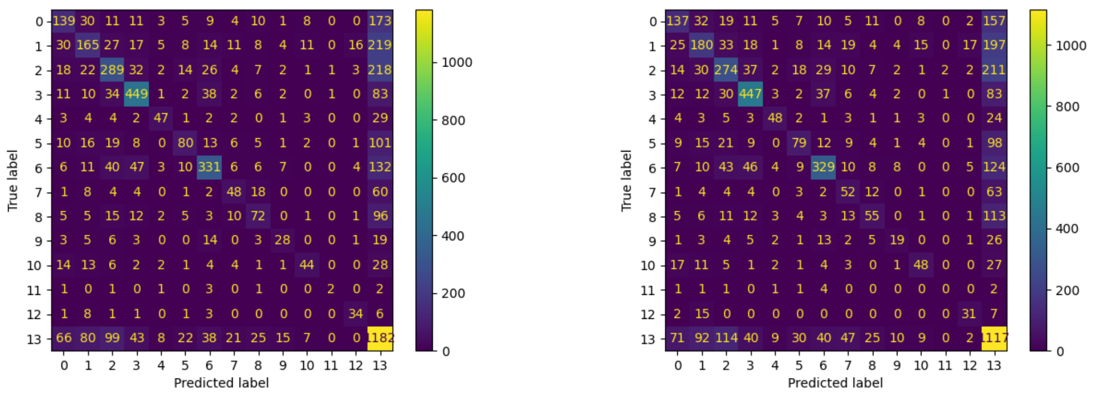


Figure 19: Confusion matrix for biGRU on 20th and 50th epoch

7 Results

Overall, I found the results surprising, my initial guess was that the models with high embedding dimensionality would have overall higher accuracy, next, I believed that if the embeddings are pre-trained, models will almost certainly produce better results, and finally, the longer the model is training, the greater results it will yield. However, the majority of my assumptions were incorrect.

Firstly, data pre-processing is of greater importance than I believed. If data is not pre-processed correctly, no matter what model is being used and how well the parameters are optimised, it is very unlikely that the model will achieve high accuracy. In this particular work, all the models were using only lemmatised tokens. However, data variability and different pre-processing techniques are very important. If I was going to redo this work, I would spend more time on pre-processing trying to understand how to feed models with features such as POS tagging.

Secondly, simple LSTM models were investigated, which ended up producing the best results. GloVe embeddings were used and investigated, however, apart from training for longer, models haven't performed any better and their confusion matrix remains slightly worse - in terms of misclassifying labels.

Thirdly, hidden dimensions and layer size play a crucial role in tunning the model and finding the optimal size - the number of parameters needed for producing good results, however, it is not a linear increase, and after a certain size, the models' performance starts decreasing. Therefore, it is important to find the balance between the size of the model - the dimensionality of the problem, and the model's performance.

Finally, although the dataset used in this work - GoEmotions is relatively small, the bidirectional GRU model does not perform better than the simple LSTMs, this could be due to several reasons. One such reason is that the dataset is simply too big for the biGRU and more complex models are needed or simply the data-preprocessing was too simple and more features should've been included.

In conclusion, the highest accuracy was achieved using an LSTM model trained from scratch, using a hidden dim of 256, embedding size of 100, a number of layers of LSTM 3 and a dropout rate of 0.5, one-cycle learning rate was used as well to increase the model's performance. Further steps that could be taken in order to increase the performance is the investigation of the attention mechanism which was introduced by the paper 'Attention is all you need'.

8 Evaluation

This section touches upon the overall attempt and outcome of this project. The challenge of having NLP models which could interact with humans and perform on-the-human level was trying to be solved for decades, and only recently have transformers made a significant step in the research community and NLP in general.

There are several applications in which models such as those built in this project could be used. For instance, there is a lot of hate happening online on various platforms. The dataset we used was scraped from Reddit comments and could be used, for instance, for detecting speech of hate (expressing anger). These applications are essential nowadays, as there are thousands of comments being posted every minute and it would not be feasible for someone to manually go through each comment and classify if a comment should be taken down or not. One can conclude that such models would have a great business perspective as today we are mostly talking and developing algorithms in other to automate the workload, and it is not only Reddit, but also other platforms like Twitter, Facebook, Instagram etc.

If one would like to consider if the models proposed in this work are sufficient, in other words, accurate enough to be used in a similar manner, further details ought to be investigated. For instance, this dataset and the models developed could be used for analysing online platform comments. Each comment could be classified as one of the 14 labels, and from that, a general understanding of users thinking could be derived. Let's assume that overall the highest achieved accuracy is 56%. If that model was implemented and certain labels were used as 'flags' for classifying speech of hate, on average, 56 out of 100 comments would be correctly classified. In my honest opinion, models trained in this work are inadequate for such purposes. In other words, it can be said that for each comment, once the labels are narrowed down to 14 labels in total, there is just a slightly higher than 50% chance that the label will be classified correctly. This is much higher accuracy than randomly guessed, as it is not a binary problem, however, in my opinion, it is not good enough to serve a business.

There are several reasons which I think contribute to this performance. In data analysis, the research expression 'garbage in garbage out' is often used, which means that depending on what we feed the models with, i.e. the data that we provide, affect the models' performance. Firstly, in this work, for evaluating models in experiments 3 and 4 only lemmatised words were used. Lemmatization can be helpful in improving the overall performance of models, however, it may not be sufficient enough for achieving high performance. For example, in experiment 1 POS tagging was used to visually display differences between certain classes. Such data is of great importance and should be included as additional features to the training algorithms. Therefore, a combination of various data pre-processing techniques and strategies is crucial to achieve sufficient performance of models.

Furthermore, various LSTM models were investigated, however, most of the changes haven't yielded significant improvement in model accuracy. Therefore, I decided to investigate pre-trained embeddings (GloVe) and test their performance, however, surprisingly, their performance almost remained the same compared to the best-evaluated model using embedding trained from scratch. In my opinion, from this one can conclude that the embedding layer had no impact on the models' performance, but rather there must be some other factors which are limiting the models' accuracy.

Finally, another factor crucial to consider is the models, one can ask themselves if these models used in this work are even powerful enough to be able to obtain all the necessary information for classifying text, and if they are simply too fragile for the data that we are working with. Several layers were investigated as well as the depth size, which had a great impact on the models performance - compared to the embedding dimensions which were not that significant. In this work, classical methods were only used, LSTMs and GRUs, however, the main breakthrough finding in NLP happened with the proposal of attention mechanism in 2017. Thus, to better understand the problem and limitations of the models, the attention mechanism should be investigated and tried implemented to the LSTMs - since they performed the best.

In conclusion, the models developed in this work are not accurate enough to be used for any business perspective using the given dataset. To be able to achieved higher accuracy attention mechanisms should be investigated, as well as the understanding of misclassified examples - trying to understand the pattern behind the errors.

Bibliography

[1] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "Goemotions: A dataset of fine-grained emotions," *arXiv preprint arXiv:2005.00547*, 2020.

[2] S. Loria, "textblob documentation," *Release 0.15*, vol. 2, 2018.

[3] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, 2002.

[4] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[5] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/4/150

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[7] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.