Text Mining and Natural Language Processing

2023-2024

[NTEGANO Bahenda Yvon Dylan, 515657]

[Emotion Recognition]

Introduction

The project focuses on text classification, specifically multiclass classification and sentiment analysis. The primary task is to classify text samples into different emotion labels.

We explore many methods for embedding the text data, starting with TF-IDF (Term Frequency-Inverse Document Frequency), followed by FastText word embeddings, and finally utilizing BERT (Bidirectional Encoder Representations from Transformers) with the uncased model. For each embedding method, we train a logistic regression model and evaluate its performance to determine the most effective approach.

The project aims to identify the best combination of embedding and classification techniques to achieve high accuracy in sentiment analysis. By comparing the results of logistic regression models trained with different embeddings, we strive to enhance the performance of text classification in identifying various emotions.

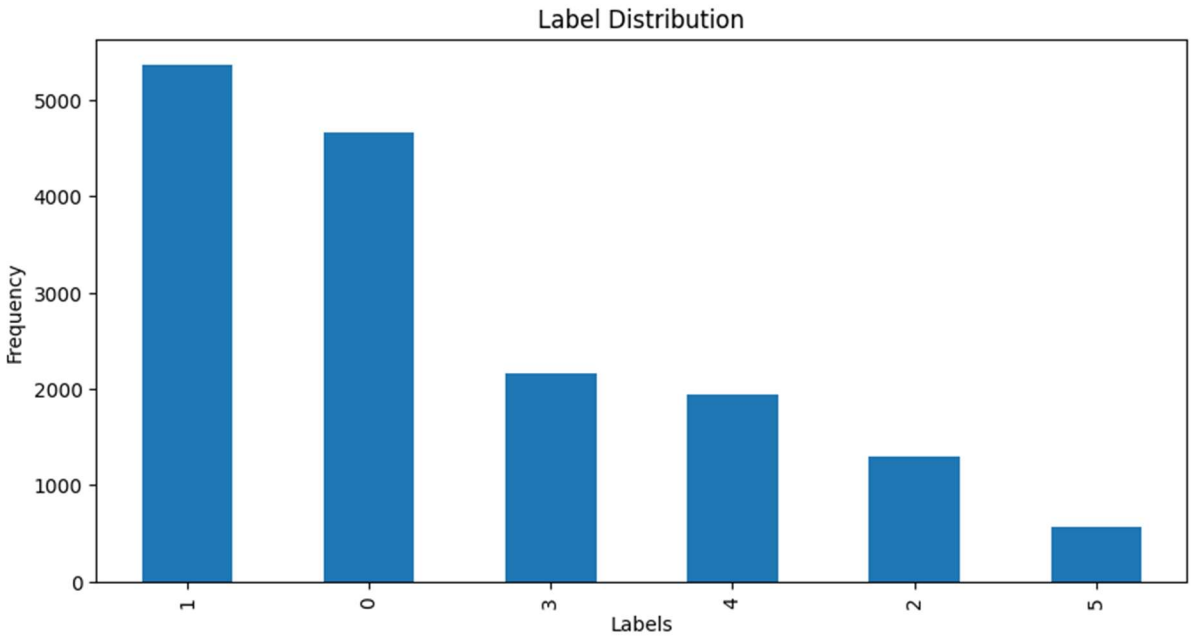
Data

The dataset used in this project consists of English twitter messages labeled with different emotions, namely sadness (0), joy (1), love (2), anger (3), fear (4) and surprise (5). The dataset is divided into training, validation and testing sets, each containing a variety of sentences representing different emotional states.

The data fields of the dataset are *text* which is a string feature that contains the messages and *label* which contains the actual labels. Here are some basic statistics:

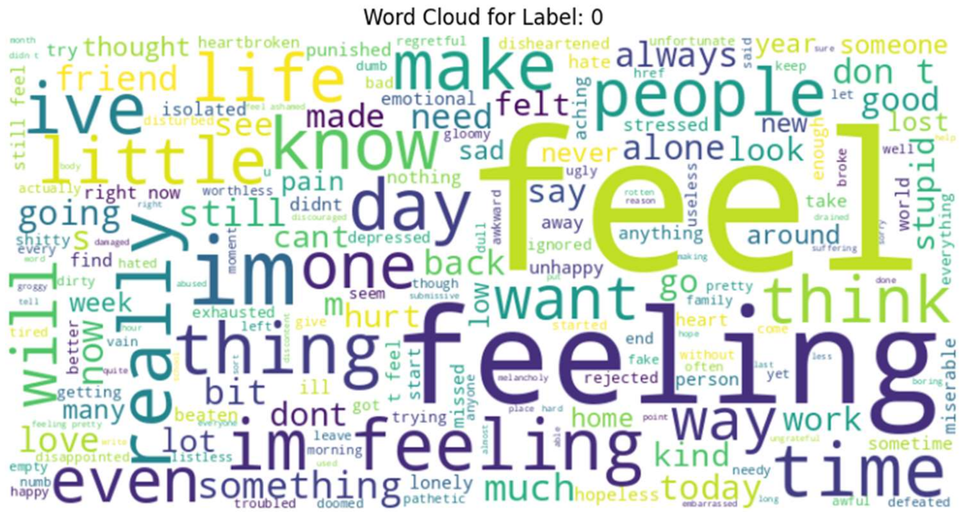
- Number of sentences for train: `16000`
- Number of sentences for validation: `2000`
- Number of sentences for test: `2000`
- Average sentence length: `19.17` words

The label distribution shows the number of samples for each emotion label in the dataset. The bar chart below illustrates this distribution.

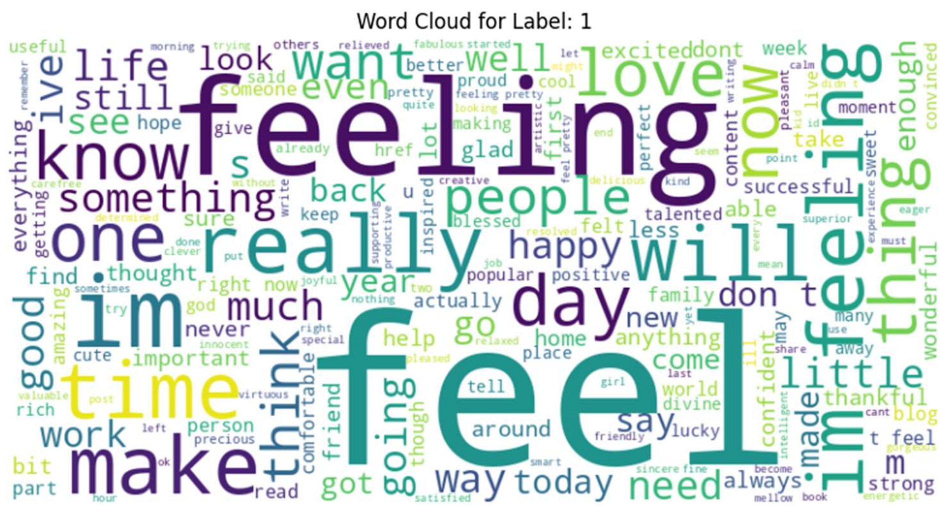


The word clouds will provide a visual representation of the most common words for each label. Below are the word clouds for each emotion label.

- Sadness


Word Cloud for Label: 0
feel
feeling
im feeling
im
really
one
thing
even
time
way
want
think
life
know
make
people
little
ive
day
will
still
going
something
much
today
friend
thought
always
don t
good
lost
alone
felt
need
made
hurt
pain
stupid
work
kind
around
back
love
lot
dont
bit
home
miserable
hopeless

- Joy


Word Cloud for Label: 1
feeling
feel
im feeling
really
will
know
one
im
time
make
day
love
want
people
think
going
life
still
something
happy
good
little
thing
now
way
today
need
much
back
work
well
excited
dont
always
confident
thankful
successful

- Love

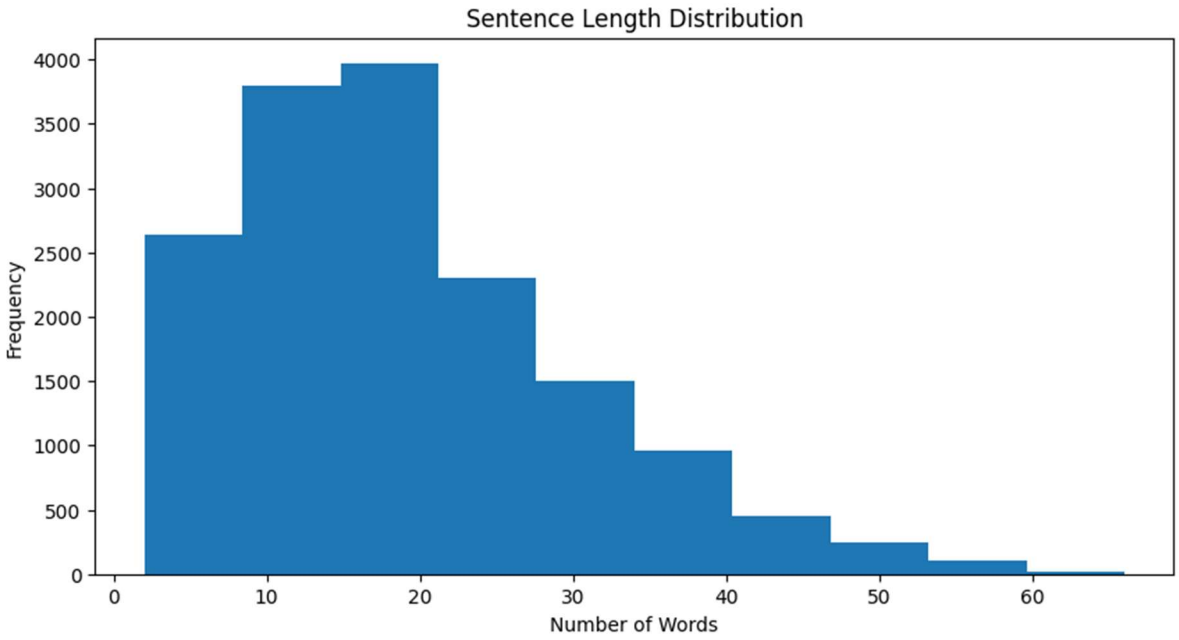

Word Cloud for Label: 2
feel
feeling
im
one
want
make
caring
will
know
really
liked
loving
lovely
time
people
sweet
life
love
hot
need
naughty
im feeling
supportive
blessed
still
romantic
supporting
little
horny
longing
think
friend
tender
gentle
fond
loyal
loved
always
faithful
passionate
delicate
nostalgic
beloved
sympathetic

- Anger

- Fear

- Surprise

The sentence length distribution shows the frequency of sentences with different lengths in the dataset. The histogram below illustrates this distribution.



Methodology

The goal of our project is to convert the text data into embeddings and evaluate a classification model using each type of the word embeddings to determine which method yields the best classification results.

Data Preprocessing

To make the data more suitable for modeling, we perform several preprocessing steps:

- **Normalization:** We lowercase all sentences, to make sure the sentences are uniform.
- **Tokenization:** We split the sentences into individual words, to make them more manageable.
- **Lemmatization:** We reduce the words to their base form; it can help in grouping together different forms of the same word.
- **Stop word and Punctuation Removal:** Punctuation marks and common words which do not contribute much to the meaning are removed to focus on the more meaningful words.

Embedding methods

1. TF-IDF (Term Frequency-Inverse Document Frequency)

It evaluates the importance of a word in a document relative to a collection of documents.

We transform the text into a sparse matrix where each row represents a document, and each column represents a word. The value in each cell of the matrix is the TF-IDF score of the word in the document. Higher scores indicate greater importance.

2. FastText

It represents words as dense vectors in a continuous vector space. It exploits character level information.

The text is tokenized, and each word is converted into a dense vector. Words are represented as vectors in a high-dimensional space (in our case, 300 dimensions). These vectors capture semantic similarities between words.

3. BERT (Bidirectional Encoder Representations from Transformers)

It generates contextual embeddings for text. It captures complex relationships between words by considering the entire context of a sentence.

The text is tokenized into subword units, and these tokens are fed into the BERT model. It produces a set of embeddings for each token in the input text. These embeddings are context-dependent and have rich semantic information.

For each embedding technique, apart from BERT which the metrics computation can be included in its model, we train a logistic regression model to classify the text samples into emotion labels. The performance of each model is evaluated using metrics like accuracy and precision. By comparing these metrics, we determine which embedding method provides more reliable and accurate classification results.

Result and Analysis

The different classification results are summarized in the table below:

Embedding Technique	Accuracy	Precision	Recall	F1 Score
TF-IDF	*0.868*	*0.868*	*0.868*	-
FastText	*0.617*	*0.640*	*0.617*	-
BERT (Eval)	*0.943*	-	-	*0.943*
BERT (Test)	*0.9215*	-	-	*0.922*

From the table, we observe a few things:

- **TF-IDF:** The model achieved a high accuracy of *0.868*, with precision and recall at the same values. This tells us that TF-IDF provides a solid baseline for text classification, capturing important word features effectively.
- **FastText:** This resulted in a lower performance, with an accuracy of *0.617*, precision of *0.640*, and recall of *0.617*. FastText captures semantic similarities between words better than traditional methods but did not perform as well in this dataset.
- **BERT:** This model significantly outperformed the other two, with an evaluation accuracy and F1 score of *0.943* and a test accuracy of *0.9215* with an F1 score of *0.922*. BERT's ability to generate rich, context-dependent embeddings from the entire sentence is likely the reason for its superior performance.

We give the BERT model, that was the best performing, some examples to see if it predicts the labels correctly. The examples given were:

1. "I can't believe my dog passed away." -Sadness
2. "I feel so happy at home!" -Joy
3. "I cherish every moment with you." -Love
4. "He yelled at me for no reason." -Anger
5. "I'm afraid of walking alone at night." -Fear
6. "I'm really shocked by the news!" -Surprise

BERT managed to predict accurately the labels of each of the examples.

We can tell from the results that even though TF-IDF provides a solid baseline with high accuracy, it lacks the capability to capture deeper semantic relationships between words.

FastText, albeit effective in some scenarios, performed lower in this dataset. The poor performance could be attributed to several reasons: it lacks ability to capture the full sentence context, it also struggles with smaller or less diverse datasets, and it is less equipped to handle nuanced sentiment understanding.

BERT, with its context- aware embeddings, significantly outperformed the other methods.

Conclusion

After installing and importing all the necessary libraries, we loaded the dataset and then we proceeded to perform some exploratory data analysis. We provided detailed descriptions of the dataset, including basic statistics, label distribution, word clouds and sentence length distribution.

The next step was to perform various preprocessing steps, including tokenization, lemmatization, normalization, stop word removal and punctuation removal.

Then we went to try different embedding techniques to see which worked better. We started with tf-idf, then we used FastText and lastly, we used BERT. For each set of embeddings, we trained a classification model and evaluated its performance using metrics such as accuracy, precision, recall, and F1 score. The BERT model had a higher performance, which points out how important contextual embeddings are.

The most challenging step was understanding and implementing the different embedding techniques, particularly BERT.

In conclusion, this project demonstrated the importance of using advanced embedding techniques for text classification tasks. Among the different techniques evaluated, FastText embeddings resulted in the worst performance, indicating limitations in capturing deep contextual information. On the other hand, the BERT model proved to be the most effective approach for accurately classifying emotions in text. The higher performance of BERT shows its ability to understand and represent the nuances of language.

AI policies

In this project, I used AI to enhance its efficiency and quality. Below are the specific prompts used:

Prompts Used:

- **"**I am doing a text mining and natural language processing project on text classification. I got the emotions dataset from hugging face. Could I do anything to improve as in data visualization and analysis "?
  In this case it was not helpful as it gave me text length distribution, label distribution and word clouds which I had already implemented.

- "Take into account the code I have written in the notebook uploaded, never mind running it, can you suggest other explorative data analysis techniques that gives meaningful information like text length distribution, word clouds and label distribution."
  When asked for more explorative data analysis techniques, it gave me interesting techniques like part of speech tagging, which I deemed unnecessary for my task.

- "Given the text classification task that I am doing, what would be a suitable representation for the text."
  It gives me different options like bag of words, tf-idf, word embeddings, subword embeddings, contextualized word embeddings and word2vec. I then chose three representations out of these.

Also used AI assistance to create the examples for manually testing the BERT model, and also for debugging, mostly errors in importing transformers.