Emotion Recognition

Text Mining and Natural Language Processing

NTEGANO Bahenda Yvon Dylan - 515657

2023-2024

Introduction

Project Overview

Multiclass text classification and sentiment analysis for emotion recognition.

Primary Task

Classify text samples into different emotion labels.

Embedding Methods

TF-IDF, FastText, and BERT explored for text representation.

Project Aim

Identify best combination of embedding and classification techniques for high accuracy.

Data Overview

Dataset

English Twitter messages labeled with different emotions.

Emotion Labels

Sadness (0), Joy (1), Love (2), Anger (3), Fear (4), Surprise (5).

Data Split

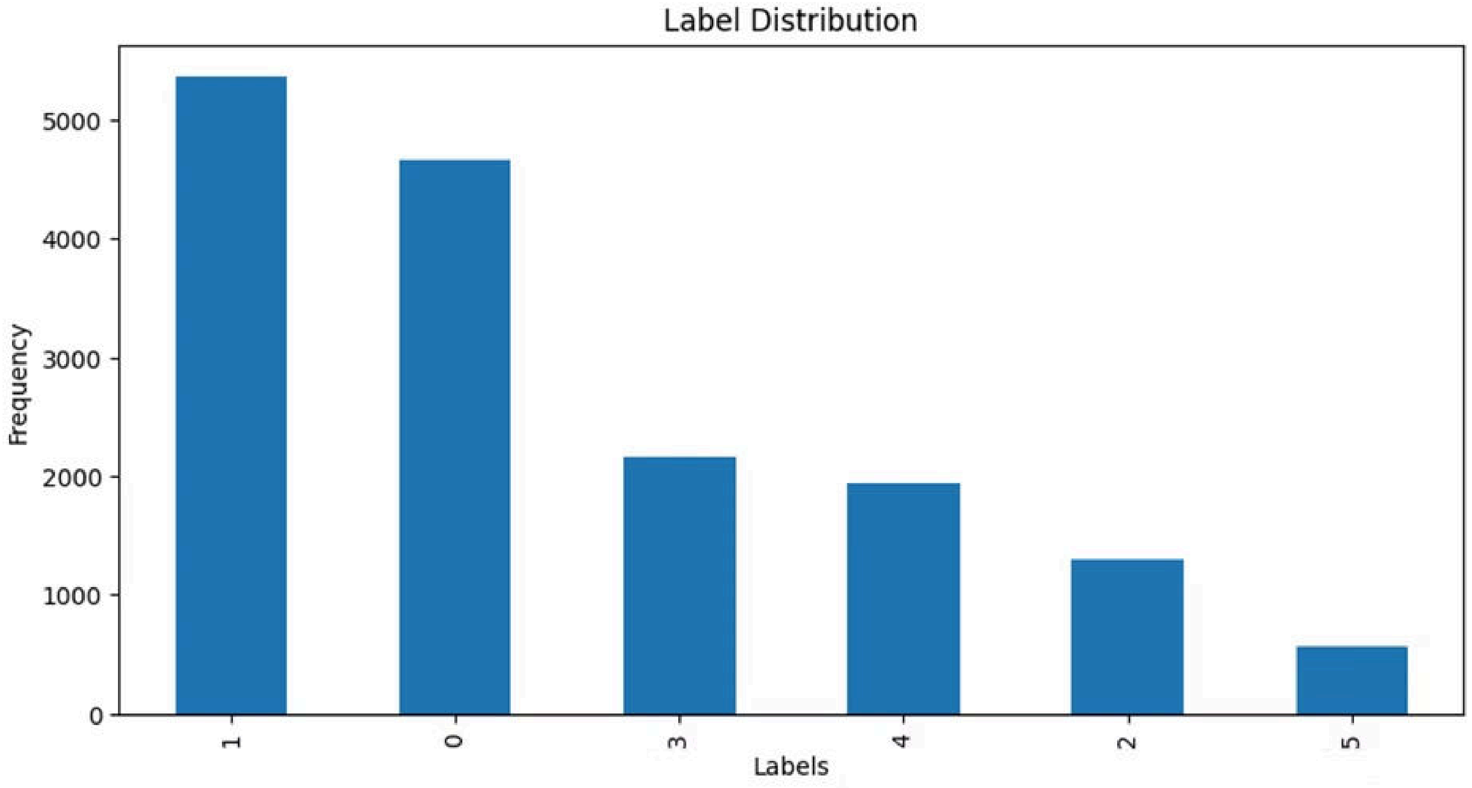
- Training: 16,000 sentences
- Validation: 2,000 sentences
- Testing: 2,000 sentences

Average Sentence Length

19.17 words

Label Distribution

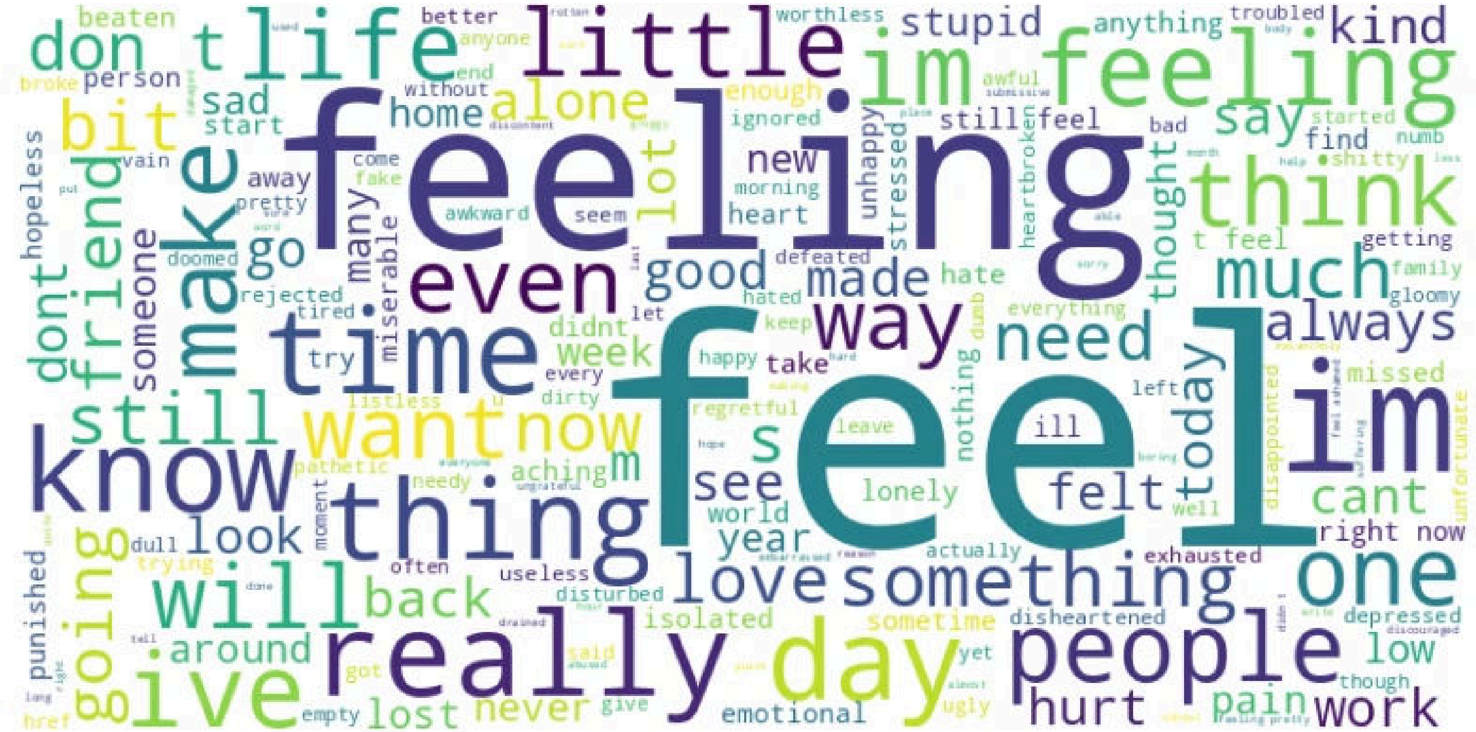
The label distribution shows the number of samples for each emotion label in the dataset.



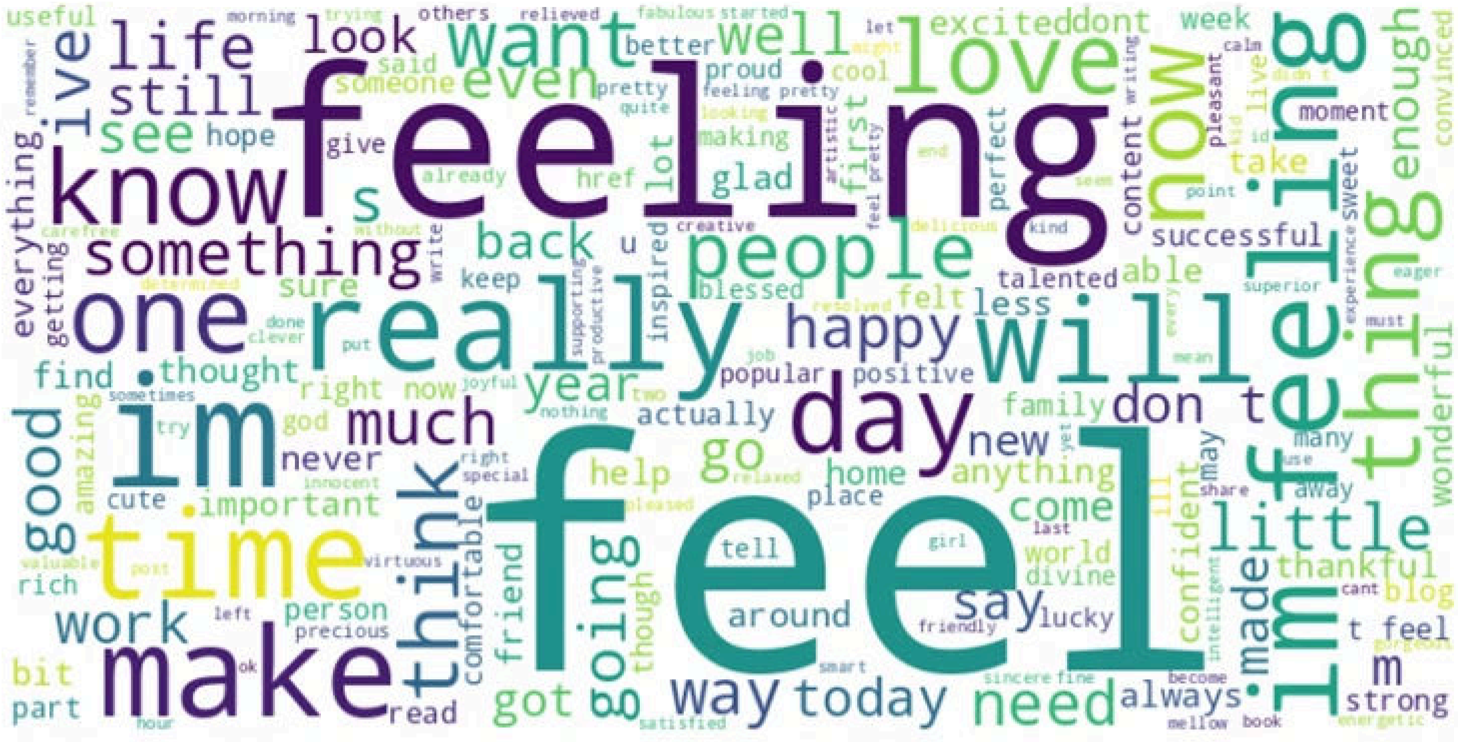
Data Visualization

Word clouds for each emotion label provide a visual representation of the most common words.

Word cloud for Sadness



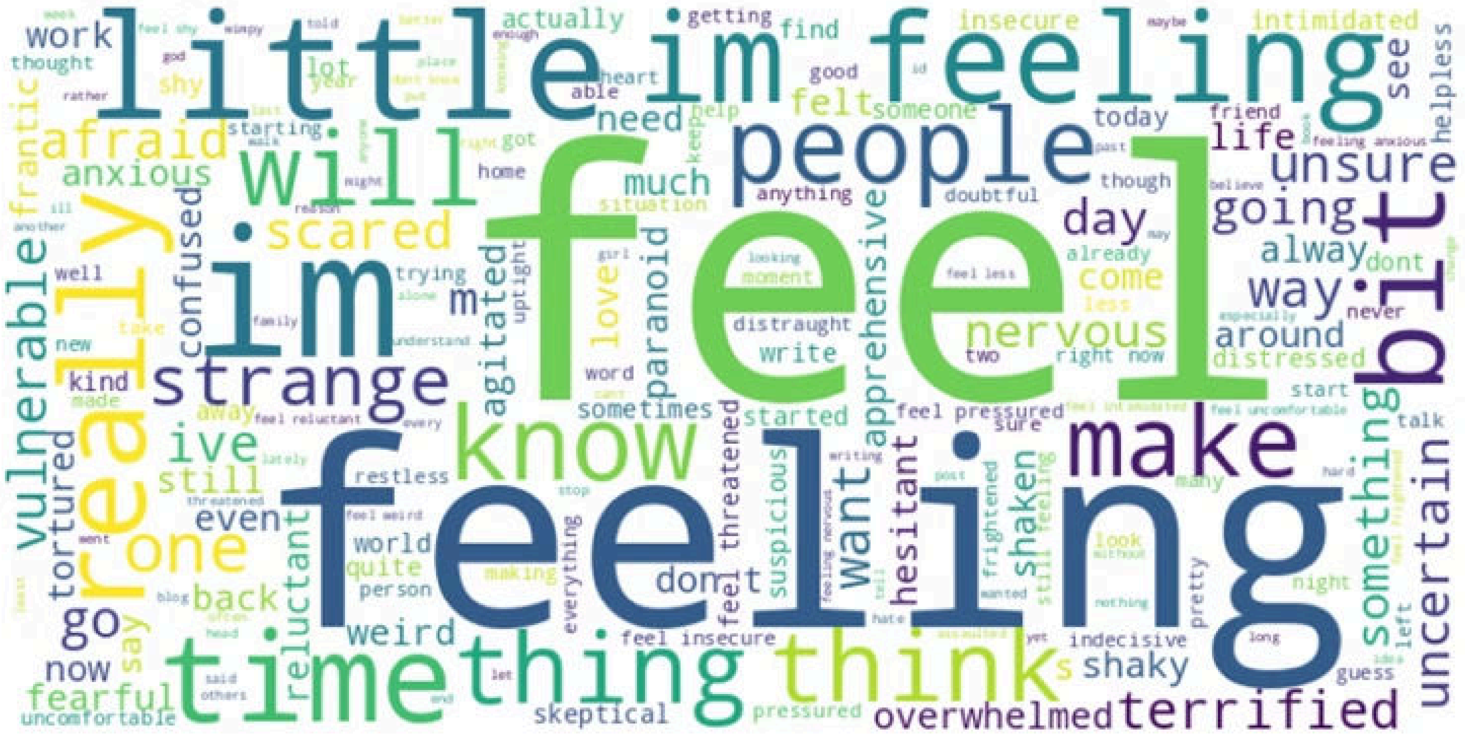
Word cloud for Joy



Word cloud for Love

Word cloud for Anger

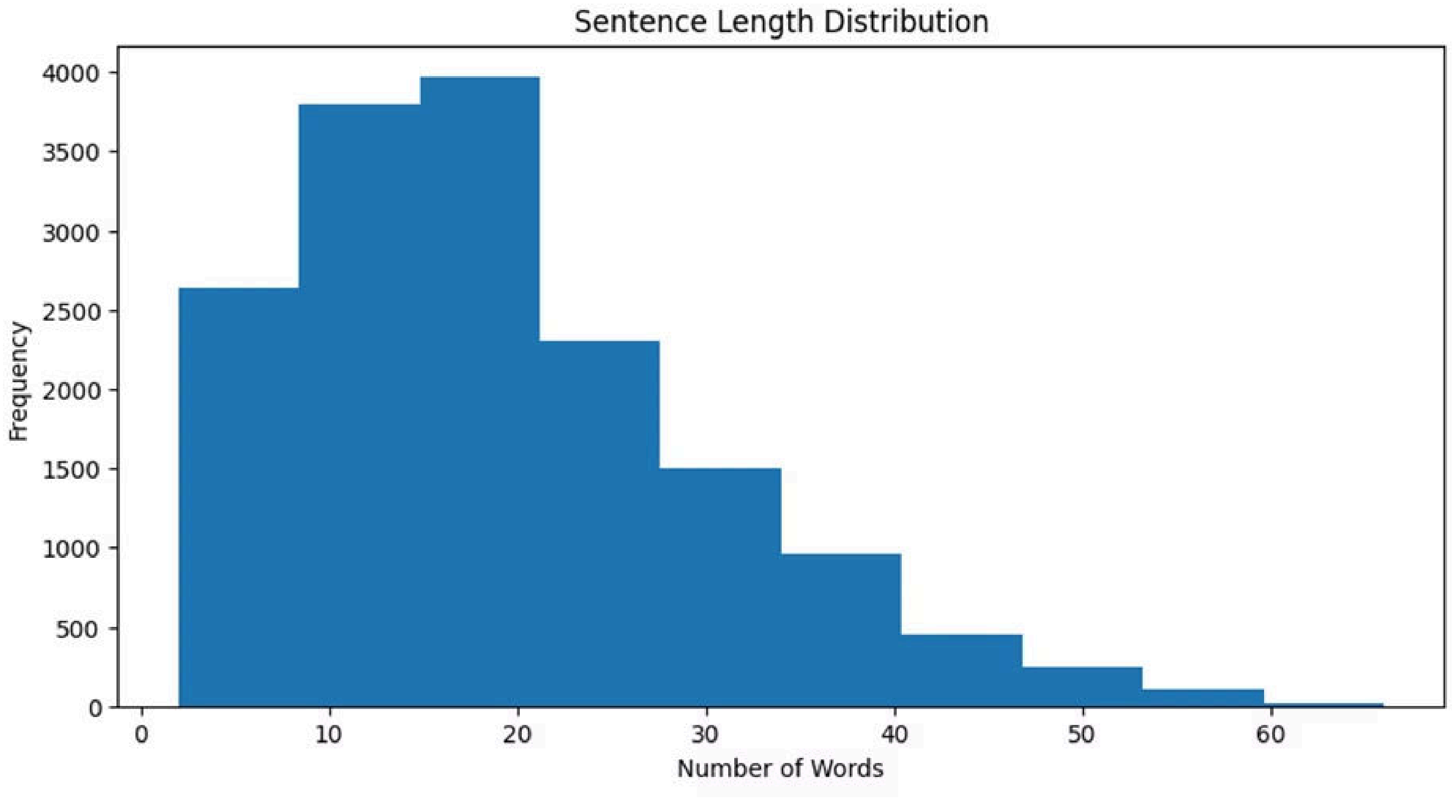
Word cloud for Fear



Word cloud for Surprise

Sentence Length Distribution

The histogram shows the frequency of sentences with different lengths in the dataset.



Methodology

- **Goal:** Convert text data into embeddings and evaluate classification models.
- **Preprocessing:** Normalize, tokenize, lemmatize, and remove stop words and punctuation.
- **Embedding Methods:** TF-IDF, FastText, and BERT.
- **Evaluation:** Assess classification model performance on embedded text.

Preprocessing the Text Data

- **Normalization**: Converting all text to lowercase.
- **Tokenization**: Splitting the text into individual words or tokens that can be processed by the machine learning models.
- **Lemmatization**: Reducing words to their base or dictionary form to group related words and reduce the vocabulary size.
- **Stop Word Removal**: Eliminating common words like "the", "a", and "is" that don't carry significant meaning for the emotion classification task.

These preprocessing steps help to clean and transform the text data into a format that can be effectively used by the machine learning models for emotion recognition.

Example preprocessing code snippet

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text) # Remove numbers
    text = re.sub(r'[^\w\s]', '', text) # Remove punctuation
    words = word_tokenize(text)
    words = [word for word in words if word not in stopwords.words('english')]
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(words)
```

Embedding Methods

The three text embedding methods - TF-IDF, FastText, and BERT - each have unique strengths for the emotion recognition task. We'll evaluate their performance to identify the optimal approach.

TF-IDF (Term Frequency-Inverse Document Frequency)

- Evaluates word importance in documents.
- Transforms text into a sparse matrix.

Example TF-IDF code snippet

```
from sklearn.feature_extraction.text import TfidfVectorizer

documents = ["I love this product", "This is an amazing place", "I feel great about the new job"]

vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(documents)
tfidf_array = tfidf_matrix.toarray()
print(tfidf_array)
```

FastText

- Represents words as dense vectors.
- Uses character-level information.
- Captures semantic and syntactic relationships.

Example FastText code snippet

```
import fasttext.util
fasttext.util.download_model('en', if_exists='ignore')
ft = fasttext.load_model('cc.en.300.bin')
sentence = "I love this product"
words = sentence.split()
word_vectors = [ft.get_word_vector(word) for word in words]
sentence_vector = np.mean(word_vectors, axis=0)
print(sentence_vector)
```

BERT (Bidirectional Encoder Representations from Transformers)

- Generates contextual embeddings.
- Captures complex word relationships using sentence context.

Example BERT code snippet

```
from transformers import BertTokenizer, BertModel
import torch
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
sentence = "I love this product"
inputs = tokenizer(sentence, return_tensors='pt')
outputs = model(**inputs)
hidden_states = outputs.last_hidden_state
sentence_vector = hidden_states.mean(dim=1).squeeze().detach().numpy()
print(sentence_vector)
```

Results and Analysis

The performance of the different embedding techniques and models is 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

The BERT model achieved the highest accuracy of 94.3% on the evaluation set and 92.2% on the test set. This suggests BERT's contextual representations are highly effective for the emotion recognition task, capturing the nuances of language and sentiment.

Conclusion

- Importance of advanced embedding techniques in text classification.
- **TF-IDF:** Provides a solid baseline but lacks deep semantic capture.
- **FastText:** Shows limitations in contextual understanding.
- **BERT:** Contextual embeddings significantly enhance performance.
- Project demonstrates BERT's effectiveness in classifying emotions in text.

@inproceedings{saravia-etal-2018-carer,

title = "{CARER}: Contextualized Affect Representations for Emotion Recognition",

author = "Saravia, Elvis and Liu, Hsien-Chi Toby and Huang, Yen-Hao and Wu, Junlin and Chen, Yi-Shin",

booktitle = "Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing",

month = oct # "-" # nov,

year = "2018",

address = "Brussels, Belgium",

publisher = "Association for Computational Linguistics",

url = "https://www.aclweb.org/anthology/D18-1404",

doi = "10.18653/v1/D18-1404",

pages = "3687--3697",

abstract = "Emotions are expressed in nuanced ways, which varies by collective or individual experiences, knowledge, and beliefs. Therefore, to understand emotion, as conveyed through text, a robust mechanism capable of capturing and modeling different linguistic nuances and phenomena is needed. We propose a semi-supervised, graph-based algorithm to produce rich structural descriptors which serve as the building blocks for constructing contextualized affect representations from text. The pattern-based representations are further enriched with word embeddings and evaluated through several emotion recognition tasks. Our experimental results demonstrate that the proposed method outperforms state-of-the-art techniques on emotion recognition tasks.",

}

Thank you!!