Chapter 3. Bibliographic Research

In this chapter, bibliographic research will be conducted to better understand how our project output could help people improve their mental health and well-being, by tracking their moods, activities, and emotions. We will focus mainly on emotion detection, which is a trivial task in our application.

Mental health and well-being are important aspects when it comes to individuals’s overall health because they influence their thinking, emotions, behaviors, and decision-making. These two problems have been addressed especially after the COVID-19 pandemic, when the use of digital platforms and social media increased significantly, and because people expressed themselves via those platforms, they became known as significant sources of information and communication regarding personal and communal mental conditions.

3.1 Emotion Models

Emotion models can be classified into two categories, as presented in [1]. The first one, categorical approach, involves placing emotions into categories that are universally recognized. Based on this, Paul Ekman came up with a model consisting of six base emotions: happiness, fear, anger, surprise, sadness, and disgust. Robert Plutchik proposed eight emotions expressed in bipolar pairs such as surpirse/anticipation, joy/sadness, anger/fear, trust/disgust, and have different intensities based on the perception of one’s experience [2]. The model proposed by Orthony, Clore, and Collins (OCC) differs from the previous ones, as they suggest a list of 22 emotions, 16 new emotions added to the model proposed by Ekman (like, dislike, hope, envy, relief, appreciation, self-reproach, shame, reproach, pity, admiration, disappointment, grief, gratification, fears-confirmed, gloating). The second one, the dimensional approach, considers emotions to be related to each other rather than to be independent. In this category includes Russell’s circumplex model on two dimensions, where Arousal indicates how excited/apathetic the emotion is, and the dimension Valence indicates how positive/negative an emotion is [4].

Based on these emotions models, researchers provided lexicons and corpora – a collection of linguistic data used to detect emotions from text. A well-known lexicon is WordNet-Affect, which is an extension of WordNet Domains, including a subset of synsets suitable to represent affective concepts correlated with affective words. It consists of 2874 synsets and 4787 words [3]. AFINN lexicon [17], contains a list of more than 3300 English terms manually rated for valence with an integer in the interval [-5, +5], not taking into consideration arousal and dominance or subjectivity/objectivity.

3.2 Computational approaches

Emotion detection in text can be achieved through different approaches, as presented in [5]:

1. Lexicon-based techniques

This approach uses lexicons, such as WordNet-Affect or SentiWordNet, to obtain a knowledge of essential elements associated with emotion labels. There are some important steps for data processing: stopwords are removed, tokenization, and lemmatization. Negation checks are used to identify the keywords and evaluate the intensity of emotion, in the end, sentences being labeled with an emotion label. [4]

In [7], a new data set, named Corpus-Based of Emotion (CBE) is created by merging the WordNet-Affect (WNA), which provides categorical labels for emotions, with Affective Normas for English Words (ANEW), which proves dimensional scores (Valence, Arousal, Dominance – VAD). Combining these attributes is difficult because of the different concepts and also it can lead to incomplete data. This is solved using the Automatic Tagging of incomplete data process. For this, they have to find the near neighbour of term x, using Adapted LESK method, for classification, while for the VAD score they use the 5 nodes nearest neighbor which is a complex task also. ISEAR dataset was used for training to obtain an expanded CBE using Latent Dirichlet Allocation (LDA). This paper concluded with the results for both CBE and CBE extended while performing emoton detection with F-measure equal to 0.5, and 0.61 respectively.

In [6], researchers came up with a method that labels sentences using emotions classes. They created 25 emotion classes, and to each class, keywords, proverbs, and emoticons are associated. While trying to classify a sentence, they first search for provers from where an emotion can be distinguished. If there is no proverb, they look for related keywords, from a total of 460 which are related to the emotion class. If there are such keywords, negation checking comes next, and finally, the sentence is classified to an emotion. In case no keyword is found, a message is provided. Next, they look for emoticons and short forms, which can be identified after words’ tokenization. These emoticons and short forms are collected from different social platforms. This method provides 80% correct outputs.

1. Machine Learning-based techniques

Unsupervised learning and supervised learning are the two domains in which machine learning divides; the first one uses labeled training data and the second one does not.

In [8], researchers used the dimensional emotion model for emotion classes. The goal of this research is to automatically predict the emotional state in text stream messages, so they divided it into two parts. The first one (Emotex) relies on collecting a large amount of labeled data messages from the Twitter platform and training the model offline. The second one, also divided into two tasks (EmotexStream), results in using the model for live-stream text messages (tweets) and classifying them to the corresponding emotion label. This is achieved through a binary classification between explicit emotions tweets and tweets without emotions. The first category is then classified using the model. Labeling the data for training is a complex task, so researchers achieved this through the usage of hashtags which are very common for tweets. Finally, 3 methods are used for classification: SVM, Naïve Bayes, and decision tree, and the results using F-measure and Naïve Bayes approach seemed to have the highest accuracy of 90%.

Due to the inefficient methods proposed since then, researchers evaluated in [9] different approaches of machine learning classifiers for emotion detection. In addition to [10] - a research paper that proposed a solution for emotion signal detection using only one machine learning classifier – Naïve Bayes classifier with 64-23% accuracy, this paper came up with an improved solution that implies 5 machine learning models. They used ISEAR dataset which contains 2273 reviews labeled with 5 emotions (sadness, fear, joy, guilt, shame), pre-processed it through tokenization and stop word removal, split it into training dataset – 80% and testing dataset – 20%, and applied machine learning classifiers. CountVectorizer and TF-IDF for feature extraction. CoutVectorizer encode a given text into a vector based on the frequency of each word that appears on the entire document, and TF-IDF calculates how relevant a word in a corpus is to a text. The used machine learning models are Support Vector Machine (SVM) whose performance was of 64.66% accuracy, Logistic Regression with an accuracy of 66.58%, KNN with an accuracy of 57.81 %, Naïve Bayes with an accuracy if 63.6%, Random Forest with an accuracy of 64.02%, XG Boost with an accuracy of 58.54%, SGD Classifier with an accuracy of 65.57%, and BPM with an accuracy of 71.27%. Finally they conclusioned that Back Propagation Neural classifier and Logistic Regression classifier had the best performance.

1. Deep Learning and Hybrid techniques

Deep Learning is a technique that we can say, tries to imitate the human brain in processing information or signals, using simple concepts to learn complex ideas, through several layers of interconnected neurons. Due to its parallel architecture, processing time is significantly reduced.

[11] proposes an LSTM (Long Short-Term Memory) approach to detect emotions in textual data - conversations, by combining sentiment and semantic based embeddings (using 2 layers of LSTM) so the model was named Sentiment ans Semantic LSTM (SS-LSTM). The architecture scheme shows that these two feature representations are combined and transmitted to a fully connected network, expecting for classification. Addressing the multi-class classification problem, the output is expected to perform the classification of text by four emotion classes: sad, happy, angry, others. Firstly, through a semi-automated approach, a 17.62 million tweets and tweets’ responses dataset is collected, and it contains 456k entrances for others class, 36k for angry class, 34k for sad class, and 28k for happy class, 90% is used for training and 10% for testing. After trying different methods, they chose GloVe as the embedding layer based on cross validation. For model training, the hyperparameters are set to these values: learner: Stochastic Gradient Descent, loss functions: Cross Entopy with Softmax, batch size = 4000, learning rate = 0.005. The following scores are determined: for the happy class, F1 score is equal to 59.68, for the sad class 80.79, and for the angry class 71.34. Even though the challenge of this approach comes in understanding context without facial or voice modulations, it performs better than the previously discussed approaches.

In [12], the proposed solution for emotion classification into eight classes is based on building a CNN model. This is achieved by first collecting the data used for training which consists of tweets containing emotion hashtags. After that, the embedding model is built, starting from GloVe model and through backpropagation the adjustment of neural network layers is performed, yet optimized. As a third step, on the ROCStories dataset most emotional words are detected using NLTK VADER – a sentiment analyzer, and as a final step, the classification is done using the trained model based on extracted words. The outcome of this aproach results in joy emotion having the highest accuracy of 73.3% while anger the lowest one, 36.7%. One of the disadvantages of this approach is that it does not take into account contextual information.

In [13], due to the lack of data, researchers proposed a transfer learning approach to classify emotions in Hindi language. Because when this method was proposed for text classification there was no dataset for Hindi language they created one, extracting sentences form Hindi news websites, labeling them with nine different emotions (also no emotion label), following Plutchik’s model. To solve the problem of limited resources they used resources from English language and generated a cross-lingual word embedding to map words from these two different languages in a shared vector space and different transfer lerning strategies. These embeddings are passed to Bi-Directional Long Short Term Memory (Bi-LSTM), generating a contextual representation, and these are passed to three Convolutional Neural Networks (CNN), and concatenated the max-pooled vectors of each. As a conclusion, Gradual Unfreezing (GU), one approach for transfer learning, perform the best with F1 score equal to 0.877. It is important to mention that by using the Bi-LSTM they were able to extract contectual information of data.

Recurrent Neutral Networs (RNN), including LSTM and GRU (Gated Recurrent Units), are connectionist models [14] that describe the dynamics of sequences by using computational cycles in neurat network. This approach differs from the conventional feedforward networks (CNN) by retaining a state which can reflect information from an arbitrarily long period of context. Even though RNNs usually contained millions of parameters and difficult training, large-scale learning was improved through developments in network architecture, optimization methods, and parallel processing.

Transduction models and sequence modeling now include the so-called attention mechanism [15], a component that models dependencies independent of their distance in the input/output sequences. They are combined with RNNs.

In [16], researchers proposed a method for the SemEval 2019-Task 3 commpetion for Contextual Emotion Detection in Text, which combines two deep-learning models: Bidirectional Encoder Representations for Transformers (BERT) and Hierarchical LSTMs for Contextual Emotion detection (HRCLE) models in order to classify conversation as happy, angry, sad, others. With the proposed approach they ended up on the 5th position in the classement. Firstly, for semantics and emotion encodings, they used a combination of pre-trained embeddings such as GloVe, ELMo and DeepMoji. A Hierarchical or Context recurrent encoder-decoder component is built to capture the conversational context, containing two types of RNN: encoder RNN and context RNN. BERT model is fine-tuned in a way that concatenates the first two utterances as the first sentence, and the third utterance as the second sentence of the pair, because they assumed the first 2 provide additional context information while the third one is related to emotion, considering it differently. In conclusion, the F1 score is equal to 0.7706, showing the HRCLE model has the best performance.

Transformer is a specific sort of neural network architecture that performs better in emotion detection due to its efficacity in learning the context and tracking relationships between sequence components. In [18], a comparative study is elaborated by analyzing the performance of five different models such as BERT, DistilBERT, RoBERTa, XLNET and ELECTRA in emotion detection from text task. For this, they chose the GoEmotions dataset which contains 58k Reddit comments in English, manually annotated with 28 emotion labels, one of them being neutral. The same hyperparameters were used to fine-tune each model. The results underline that RoBERTa model performed the best by f1 score = 0.49. XLNET had the longest time for training, while ELECTRA the slowest one.

Chapter 4. Analysis and Theoretical Foundation

In this chapter, we take a closer look at the detailed concepts and frameworks utilized in the development of our system designed to enhance the users’ mental health and well-being by monitoring their moods and activities and also by helping them to better understand their emotions and how they feel, through an important feature, emotion detection.

The so-called Mind Bloom Application is developed using MERN stack, MongoDB, Express.js, React.js, and Node.js. Additionally, an AI model is used, more precisely a fine-tuned RoBERTa model from Hugging Face, which detects emotions from text input, which represents the highlight of the day for one user.

4.1 Theoretical Foundation

In this section, we will elaborate on the concepts and frameworks used to develop a system able to determine a person’s emotions from an inputted text using a fine-tuned model. We will define the key terms and concepts, and explain the logic behind the chosen approach, which results in good comprehension of the Transformers library, BERT model, RoBERTa pre-trained model, MERN stack, and REST API.

After exploring and analyzing the different approaches used for emotional detection in text, in Chapter 3, deep learning techniques have proven their efficiency using RNN, the preferred approach when dealing with sequential data, by the time of the Transformers model introduction. RNNs operate similarly with a feed-forward function, but they process the input sequentially, one element at a time. The training of such recurrent models can be slow because they do not use GPUs, which allow for parallel computations. RNNs have loops in them, which allow the information to be passed from one step to the next, and thus persist. But when elements are distant from one another, the probability of information being lost increases, so they become ineffective. The encoder-decoder architecture of RNNs motivated Transformers, but they are different by being completely based on the Attention mechanism, rather than recurrence. Besides improving the performance of RNN, Transformers architecture is designed for dealing with several additional tasks such as text summarization, speech recognition, and image captioning.

4.1.1 Transfomers

A transformer is an encoder-decoder model that uses the attention mechanism, firstly introduced in [15]. In figure 4.1 is presented the overall orchitecture of Transformers. As we said, it can deal with different tasks, so from the full architecture, we can leverage only some parts of it according to what we are trying to do. To better understand the architecture we will firstly discuss about the Attention Mechanism, the innovative principle on which this architecture is based, making it possible to solve the mentioned tasks above.

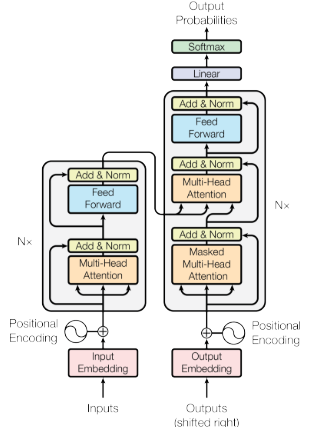


Figure 4.1 Transformer’s architecture (Source [15]).

The attention mechanism's role is to improve the encoder-decoder model for machine translation, by allowing the decoder to utilize the most relevant part of the input sequence in a dynamic manner which consists of a weighted combination of all the encoded input vectors in which the most relevant vectors have the highest weights. In other words, it has the ability to focus on important parts while ignoting the others that seem irrelevant at the time, like humans pay attention to certain aspects in conversations, this selective focus being of major importance where context is key.

The main component of attention mechanism is the scaled dot-product attention made up of three matrices of k dimension: query (Q), key (K), value (V). The formula presented in Figure 4.2 is applied to obtain the values’ weights.

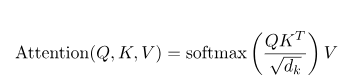


Figure 4.2 Computation of Scaled Dot-Product Attention.

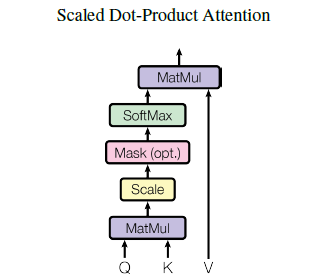


Figure 4.3 Scaled Dot-Product Attention (Source[15]).

Instead of limiting the model to learn one type of relationship, Multi-Head Attention allows to concatenate the output of multiple scaled dot-product attention, which improves the performance of the model and also its training stability, creating specific contextualized embeddings.

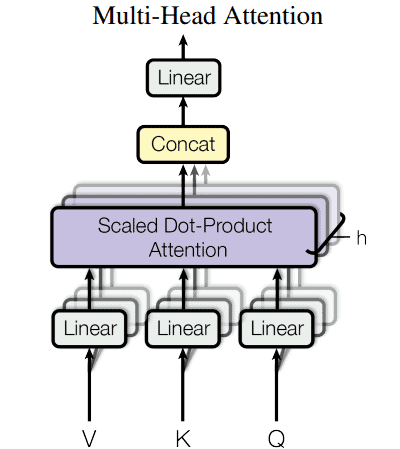


Figure 4.4 Multi-Head Attention (Source[15]).

Now that we have a good comprehension of what the attention mechanism is, we will focus on explaining the architecture presented in Figure 4.1. It is composed of two main parts: the left one is the encoder, and the right one is the decoder.

The encoder can be seen as a block that can be trained to understand an inputted sequence and extract the relevant information. It transforms the input tokens into contextualized representations with respect to the entire sequence. The first step, as can be seen in the picture, is the input embedding, which transforms each word into a vector of size 512. (fixed-size). The second step is positional encoding, which provides information about the token’s position in the sequence, resulting in embedding with positional context. The third step consists of a stack of encoder layers having the same structure but different weights. Each encoder layer is divided into two sub-layers: the multi-head attention layer about which has already been discussed and the feedforward layer which consists of two linear transformation layers, with a Relu activation in between. The output of the encoder is a set of vectors that is transmitted to the decoder, indicating it to pay attention to the right words in the input when decoding.

The decoder’s role is to decode the numerical representation output of the encoder. Its architecture is similar to the encoder, with a feed-forward layer and, two multi-head attention layers. One of the is the self-attention layer, and the other one is the encoder-decoder attention layer which helps the decoder focus on the relevant parts if the input sequence.

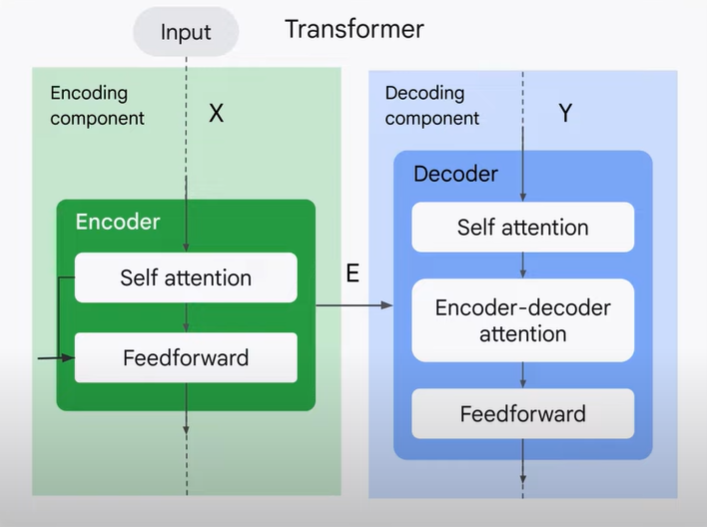


Figure 4.5 Transformer’s Encoder and Decoder structure.

Based on Transformers, several models were developed which are then fine-tuned to specific tasks in natural language processing, including emotion detection. Some of the well-known models include BERT, GPT, LaMDA. Models follow the architecture presented in Figure 4.6. Each model in the library is determined by three building blocks:

1. Tokenizer – converts raw text to sparse index encodings
2. Transformer – transforms sparse indices to contextual embeddings
3. Head – uses contextual embeddings to make task-specific predictions

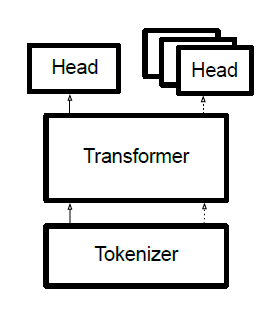


Figure 4.6 Model’s architecture (Source [19]).

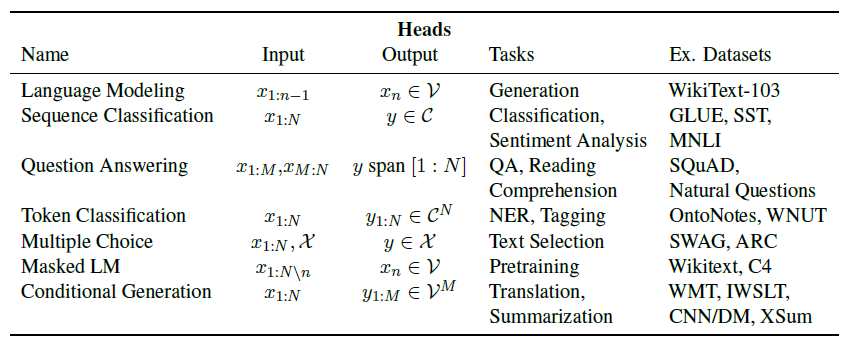


Table 4.1 Heads contextualized embeddigs based on specific tasks.

For our system, we have chosen the BERT direction approach, more exacly the RoBERTA model which is a BERT optimization, due to its performance and capability to understand the context which is trivial in emotion detection.

4.1.2 BERT model

BERT (Bidirectional Encoder Representations from Transformers) is a transformer model released by Google in 2018, which revolutionized NLP with its unique bidirectional training. This feature enables the model to have better informed context based on each predictions are done.

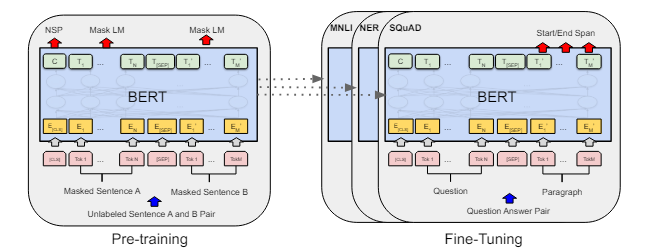


Figure 4.7 Procedures for Pre-training and Fine-tuning (Source [20]).

BERT employs an encoder only architecture and is based on two unsupervised training strategies: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) (Figure 4.7). The MLM mechanism replaces 15% of the words in each sequence with a [MASK] token which results in training the model to make predictions of the mask on the un-masked words from sequence, by adding a classification layer on top of the encoder output. The NSP mechanism gives model the ability to predict a second sentence in a pair of two is the subsequent pair in the original document, an ability which helps in maintaining a long-distance relationship between texts. This can be achieved through input processing by inserting two tokens at the beginning and at the start of the sentence, [CLS] and [SEP] respectively as ahown in the Figure 4.8.

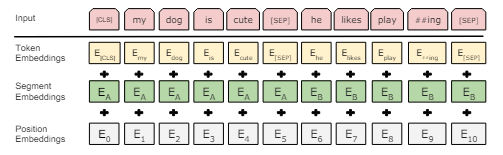


Figure 4.8 Input representation for BERT model (Source [20]).

In emotion detection, a classification task, a classification layer is added on top of the Transformer output for the [CLS] token.

4.1.3 RoBERTa model

RoBERTa model takes the core principles of BERT and optimizes them (Figure 4.9). By refining the training process the model achieves better performance, but with the cost of more computational resources. It is used in natural language processing projects including text classification, sentiment analysis, question-answering, and language modeling. In papers [18] and [22], after applying different models for emotion detection on different databeses, RoBERTa model performs the best in terms of accuracy.

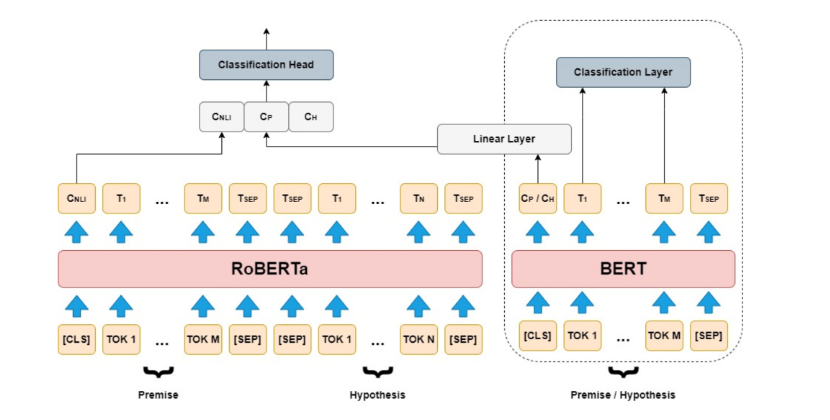


Figure 4.9 RoBERTa model versus BERT model.

In comparison with BERT, RoBERTa has been trained on 10x much more data, over 160 million phrases. It contains 24 transformers layers, while BERT only 12.

As we mentioned, during the pre-training BERT model uses language modeling by trying to predict 15% tokens, the masked tokens. The problem here is that selected tokens for masking are occasionally the same for the a text sequence across several batches.

In [23] is mentioned that for BERT the training set is replicated 10 times, which means that each sequence is masked in just 10 different ways, 40 training epochs are run, so four times each sequence with the same masking passes to BERT. To overcome this, dynamic masking strategy is applied in BERT, which means that mask is generated every time a sequence is passed to BERT in a unique way, resulting in less duplicated data during training. This approach is used in RoBERTa model (Figure 4.10). Random masking various tokens in every pre-training epoch helps the model to learn manage out-of-distribution input by exposing it to a wide range of input distribution.

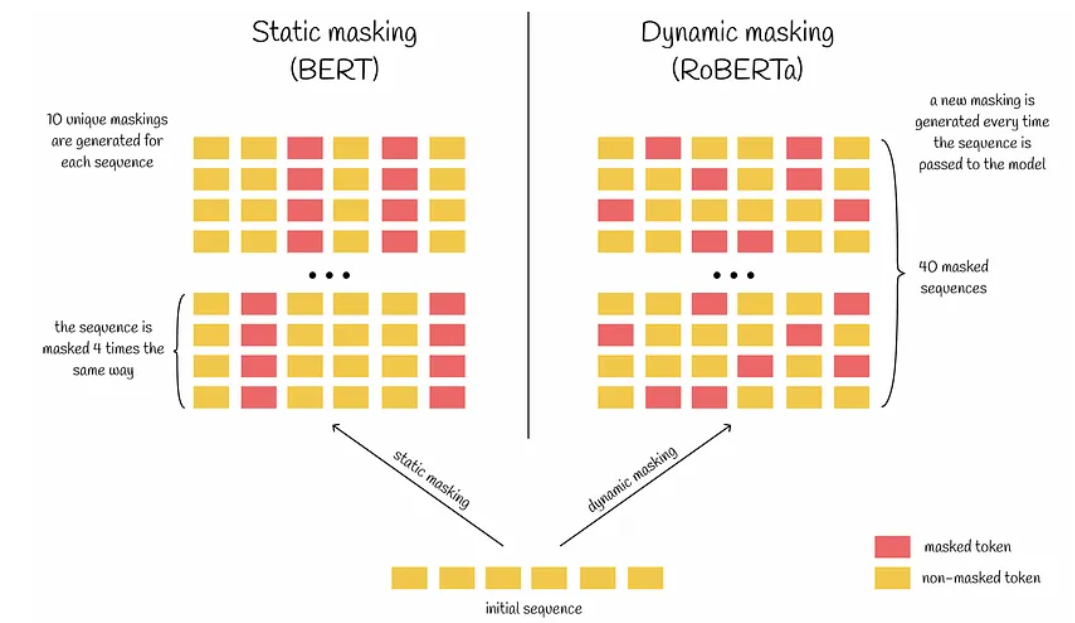


Figure 4.10. Static masking versus Dynamic masking.

In RoBERTa model the next-sentence prediction task which predicts if a given sentence is the next sentence in a text or not, is eliminated. RoBERTa is trained on complete sentences, not pair sentences.

A larger byte-pair encoding (BPE) vocabulary size is used in RoBERTa. BPE is one kind of sub-word tokenization that helps to handle better the unusual words which did not appear in the training data.

4.1.3 GoEmotions Dataset

GoEmotion is a manually annotated dataset consisting of 58k Reddit comments in the English language which are labeled with 27 emotion categories + neutral, so a total of 28 labels. There are 12 positive emotions, 11 negative emotions, and 4 ambiguous emotions, named in Table 4.2, and distribution of emotions shown in Figure 4.11.

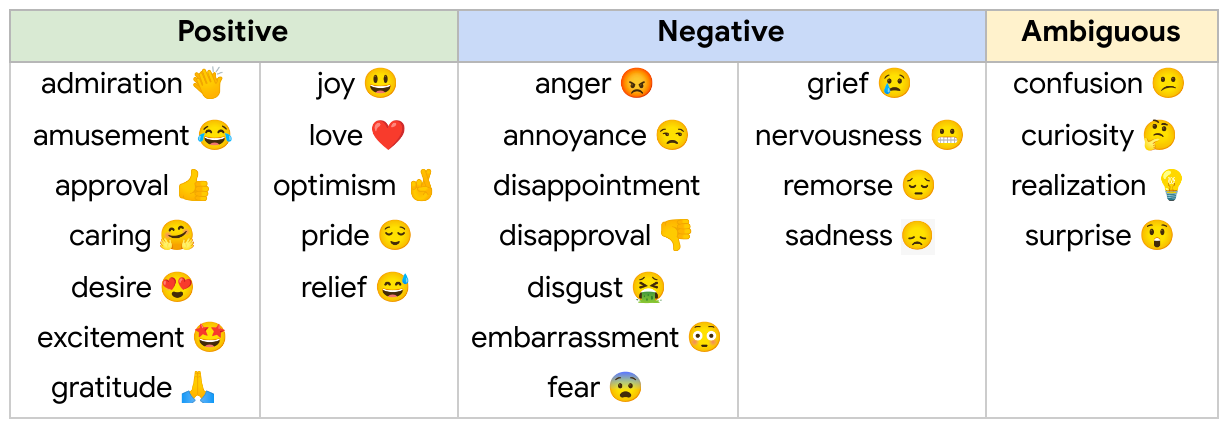


Table 4.2 Emotion labels used in GoEmotion Dataset.

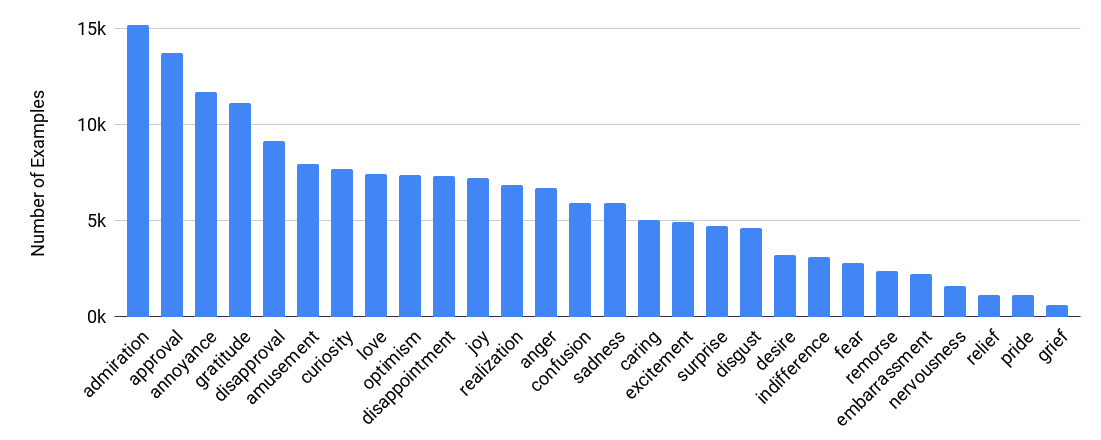


Figure 4.11 Emotion distribution in GoEmotion Dataset.

In order to have a better prediction we found o a cleaned version of the GoEmotions data set uhese

We used this dataset because it contains an emotion model of 27 emotions and because this AI component is applied to a text written by the user which represents a situation he passed through during the day, having such an emotion model would help in indicating the user that there are many emotions he can confront with, arousing his curiosity to dive deeper into the inner reflective state. MAI COMPLETEZ AICI

The AI component used for emotion detection is part of a more complex web application. The purpose of this application is to allow users to track their moods, emotions, and activities during the day to enhance well-being and mental health. To achieve this, we implemented the backend and the frontend components which facilitates the functionality and provide a user-friendly interface for our application. For this, the MERN stack was used. The technologies, advantages and disadvantages and limitations are discussed in details in the next section.

4.1.4 MERN Stack

Firstly, we will explain the concept of stack. By a technology stack we refer to a combination of compatible technologies and programming languages which are stacked together to build a fully functioning system (Figure 4.12), such as a website in our case. It consists of frameworks, backend (server-side) and frontend (cliend-side) tools, programming languages, a database, and applications linked together via APIs.

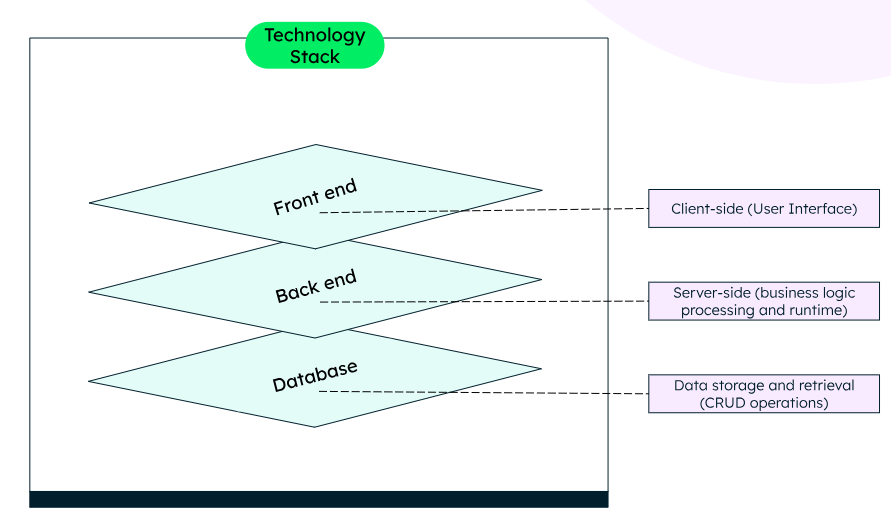


Figure 4.12 Main Components of a Technology Stack.

One of the well-known technologies stacks used in the development of a web application is the MERN Stack, a JavaScript full-stack that consists of the following technologies grouped in a three-tier architecture that interact as presented in Figure 4.13:

* M – MongoDB (Non Relational Database)
* E – Express.js (Node.js Web Server)
* R – React,js (JavaScript Frontend Framework)
* N – Node.js (JavaScript Web Server)

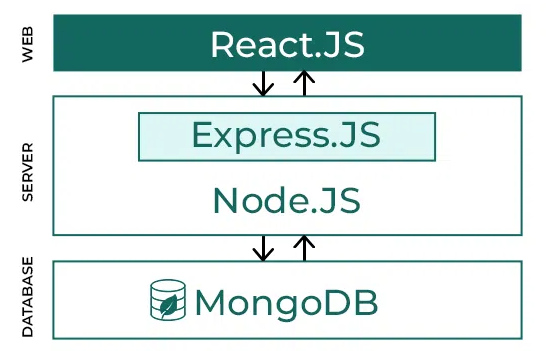


Figure 4.13 Flow between MERN Stack Components.

**React.js**

As shown in the figure 4.10, the top tier of the MERN stack (client-side), is the React component, a JavaScript library used for building dynamic client-side HTML applications, also referred to as user interfaces (UIs). This library allows us to create complex interfaces through simple components, connect them to data on backend server, and render them as HTML. It is designed as a component-based architecture, providing features to break down the UI into smaller, self-contained components with their own logic and design. It operates by creating an in-memory Document Object Model (DOM) following the Virtual DOM approach. In this way, the rendering performance is optimized by minimizing DOM updates. In React, data flows one-way, in one direction, so it is said to have one-way data binding. The data is transferred from parent components to child components, from top to bottom. It has all the features one would expect from a modern web framework offering great support for forms, error handling, events, lists, calendars, components used in MindBloom application.

**Node.js**

The second tier is the server-side where the functionality of the system is implemented. Node.js is an “open-source and cross-platform JavaScript runtime environment” [21]. It operates on a single process meaning that a Node.js app does not create a new thread for each request. Node.js has a set of asynchronous I/O primitives that prevent JavaScript code from blocking. Operations like reading from the network, which is an I/O operation, are performed by accessing a database or the filesystem. When the response come back, Node.js will resume the operations instead of blocking the thread which would waste CPU cycles waiting. This lets Node.js manage thousands of concurrent connections with a single server without needing thread concurrency, which might be a major issue cause.

**Express.js**

Express is a Node.js framework. By using this framework, developers can focus on the important part of the backend which is the business logic instead of setting up ports to route handlers and writing all the boilerplate code which can be time-consuming. At the server-side part of a web application, handling requests is mandatory. When receiving an HTTP request, the server chooses the corresponding route handler. Express.js provides methods to specify the corresponding function for a specific HTTP verb (GET, POST, PUT, PATCH, DELETE, etc.) and the route represented by a URL. Also, Express.js acts as a routing and Middleware framework for managing the different routing of the web page, working between the request and response cycle. After the server receives the request and before the controller actions send the response, middleware gets executed. It has access to the request object, responses objects, and next, it can process the request before the server sends the response. So it can be said that the middleware is a request handler that allows to intercept and control requests and responses. An Express-based application is a set of middleware function calls invoked by the Express.js routing layer. This tool helps in adding functionalities including logging, authentication, error handling, etc., (Figure 4.12).

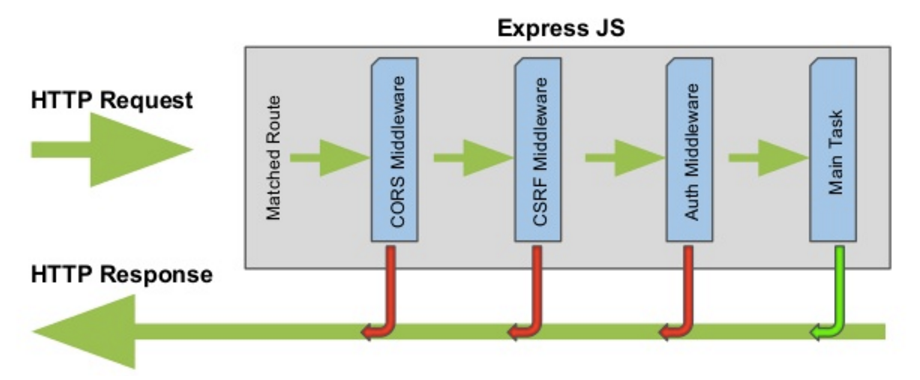


Figure 4.12 Flow diagram of middleware functions.

**MongoDB**

The last tier of the stack is the MongoDB, a NoSQL database in which every record is a document made of key-value pairs that are similar to JavaScript Object Notation (JSON), which are identified by a primary key. These documents represent the basic unit in MongoDB. These documents are stored in a collection which is analogous to tables from relational databases. NoSQL stands for Not Only SQL and is a type of Database Management System (DBMS) designed to handle unstructured and semi-structured data. NoSQL differs from the traditional traditional relational databases that store data using tables with pre-defined schemas, by offering flexible data models capable of scaling horizontally which can also adapt to changes in data structures.

It is important to mention that one of the features that makes the MERN stack work well is that the technologies save data in the same format. React stores the data as a JavaScript object, MongoDB stores data as Binary JavaScript Object Notation (BJON), the backend uses JavaScript code, and the express converts data between JS and JSON using the .json() method.

**RESTful API**

A RESTful API, also called REST API, is an architectural approach for an application programming interface that uses HTTP requests to access and use data. It conforms to the representational state transfer (REST) architectural style.

An API is a mechanism consisting of a set of definitions and protocols that allows the application to access a resource within another application. The application that accesses the resource is the client, and the application that contains that resource is the server. In other words, if our user want to retrieve information or perform an operation in our system, the API helps communicate what user wants to the system so it can undestand and fulfill the specific request. It acts as a mediator between users/clients and web services they want to get, while maintaining control, security and authentication, which determins who gets access to what.

REST is a set of architectural restrictions, not a protocol or a standard. When a user request is made via REST API, it transfers a representation of the state of the resource to the endpoint. This representation is delivered in one of several formats via HTTP, in our system as a JSON. REST is preferred because it uses less bandwidth, making it more efficient in internet use.

The main element of a REST API are:

* Client: it requests a resource from the sever
* Server: it controls the resource and responds to the client
* Resource: it is any data/content controlled by the server making it available in response to client request

The client request include four principal parts:

* HTTP methods

This method tells what is expected to happen with the specified resource. In Table 4.3 the method's verb is correlated with the action it performs.

* Endpoint

It indicates the resource’s location. It usually includes a Uniform Resource Identifier (URI), or if the resource is accessed through the internet it represents an URL address

* Header

It contains the necessary details to execute the call and handle the response, which might include authentication data, such as JWT Token, encryption key, details about the preferred data format expected for the response.

* Body

It contains relevant information to or from server. In the case of POST or PUT methods it contains the new data to be added/updated.

|  |  |
| --- | --- |
| HTTP verb | CRUD action |
| POST | Create/Insert |
| GET | Read |
| PUT | Update |
| PATCH | Update |
| DELETE | Delete |

Table 4.3 HTTP verbs correlated with actions.

**JWT Security**

JSON Web Token, commonly known s JWT is an open standard (RFC 7519) used to pass information securely between parties. The digital signature of the token assures its integrity and legitimacy. In our application, we used the JWT token for user authentication, and access to specific resources to securely access data. The KWT token consists of three parts which are separated by a dot (.): header, payload, and signature. The JWT token is used in the following way: when the user performs login, the server generates a JWT token upon a successful authentification. The client stores this token in local storage which is sent in the header of a request for each request needing authentication. By verifying the signature and payload decoding, the server validates the token and provides access to the data. (Figure 4.13).

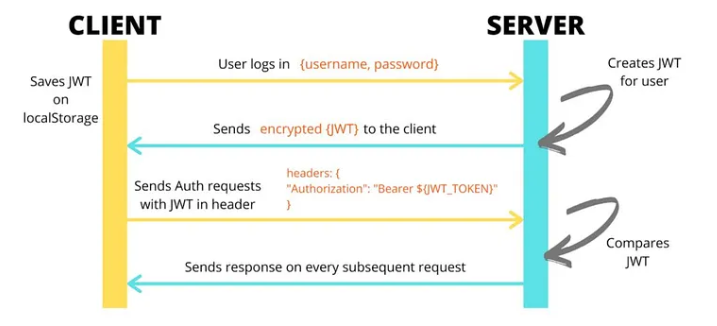


Figure 4.13 Token Based Authentication.

4.2. Analysis

In this section, we will present the main functionalities of our system, the use cases alongside visual representation to have a better overview of the functionalities in the MindBloom Web Application.

**Conceptual Architecture**

The technologies used to develop the MindBloom Web Application were briefly discussed and explained in the section 4.1 of this chapter. In the figure 4.13 is presented the Conceptual Architecture for application development based on these technologies.

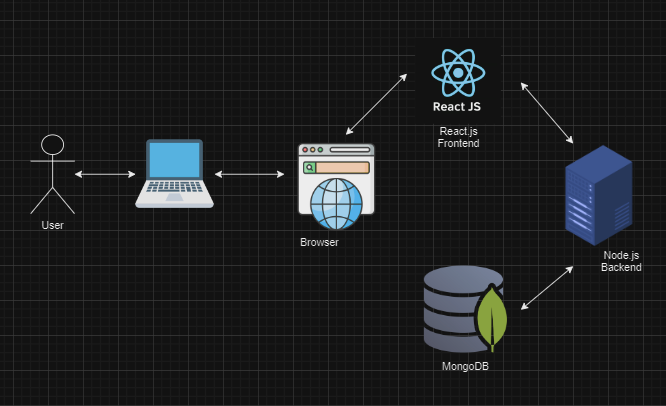


Figure 4.13 Conceptul Architecture of MindBloom Web Application

4.3 Use Cases

**Register and Login**

This is the most important use case which is mandatory for this web application. To use the application, each user must create an account, and perform login to access the resources of application. The basic flow is presenten in Figure 4.12.

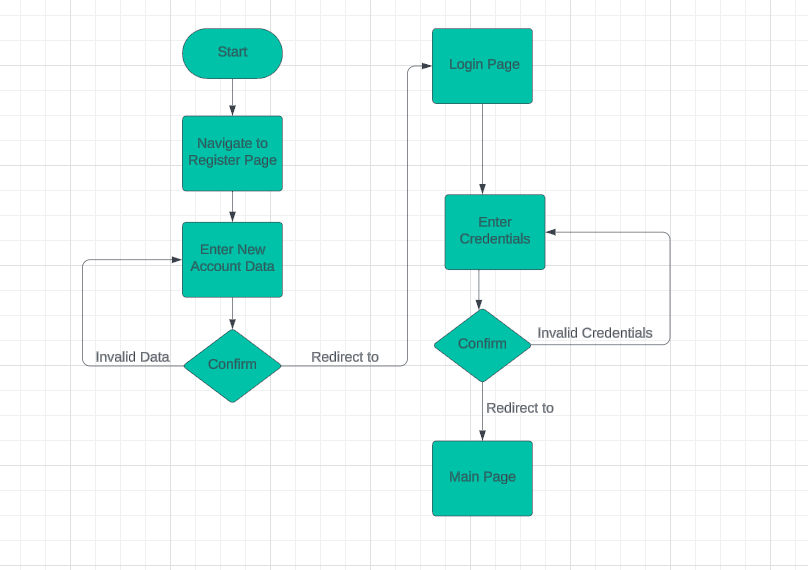


Figure 4.12 Register and Login Use Case Diagram

Use Case Start

User want to use the MindBloom Application

1. User navigates to Register Page
2. User enter the new account valid data
3. User press Register button
4. User is redirected to Login Page
5. User enters the account credentials
6. User press Login button
7. User is redirected to the Main Page

Use Case End

The user created his account and his credentials are saved in the database and now he can use the MindBloom Application.

**Add a new Entry for the day**

One important feature of the application is the Mood and Activities tracker which helps user keeps records of day-by-day life’s ups and downs. An Entry contains both moods with the associated activities. The basic flow is presented in Figure 4.13.

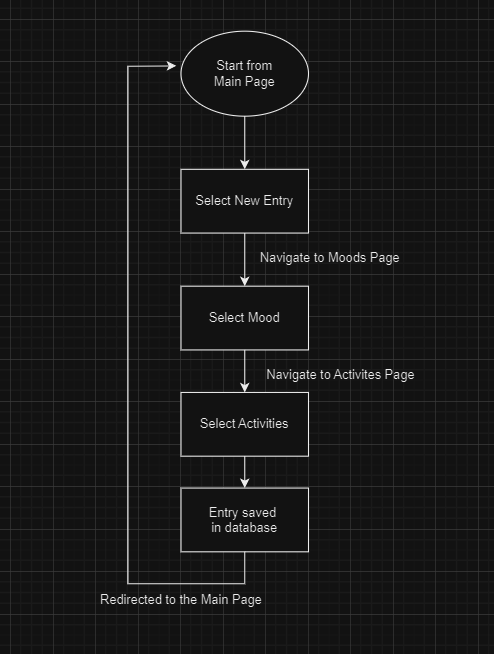


Figure 4.13 Record a new Entry

Use Case Start

User wants to record a new Entry for the day with consists of actual mood correlated with activities.

1. User presses on New Entry Button from the Options Menu
2. User selectes one mood out of five represing his actual mood
3. User presses Submit button
4. User is redirected to the Activities Page
5. User selects the activities he performed correlated with his mood
6. User presses Submit button
7. User is redirected to the main page

Use Case End

The new Entry consisting of mood and activities is saved in the databse and can be accessed by the user for visualization.

**Journals and Notes**

Another two important functionalities are the Note Page, where the user can type anything and keep it as a note, and the Journal Page where the user type a highlight of the day, based on which emotion detection is performed. This last feature was one of the most challenging one. The basic flow is presented in Figure 4.14.

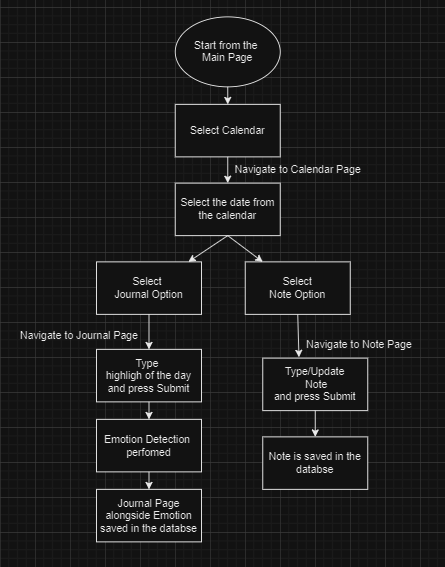


Figure 4.14. Basic flow for creating a Jounal Page or a Note Page

Use Case Start

User want to type a note or highlight of the day

1. User presses the Calendar button
2. User is redirected to the Calendar Page
3. User selects a date
4. User selects an option
5. User type the corresponding text
6. Text is saved in the databse

Use Case End

User can access anytime the Journal Page or the Note Page saved in the databse.

Bibliography:

[1] Gunes, H., & Pantic, M. (2010). Automatic, Dimensional and Continuous Emotion Recognition. International Journal of Synthetic Emotions, 1(1), 68–99. doi:10.4018/jse.2010101605

[2] Plutchik R. A general psychoevolutionary theory of emotion. Amsterdam, Netherlands: Elsevier; 1980 (pp. 3–33).

[3] Strapparava, C., & Valitutti, A. (2004, May). Wordnet affect: an affective extension of wordnet. In Lrec (Vol. 4, No. 1083-1086, p. 40).

[4] Ashritha R Murthy and K M Anil Kumar 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1110012009

#### [5] Nandwani, P., & Verma, R. (2021). A review on sentiment analysis and emotion detection from text. *Social Network Analysis and Mining*, 11(81).

#### [6] Rahman, R. (2017). Detecting emotion from text and emoticon. London Journal of Research in Computer Science and Technology.

#### [7] Rachman, F. H., Sarno, R., & Fatichah, C. (2016, October). CBE: Corpus-based of emotion for emotion detection in text document. In 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE) (pp. 331-335). IEEE.

#### [8] Hasan, M., Rundensteiner, E., & Agu, E. (2019). Automatic emotion detection in text streams by analyzing twitter data. International Journal of Data Science and Analytics, 7, 35-51.

#### [9] Asghar, M. Z., Subhan, F., Imran, M., Kundi, F. M., Shamshirband, S., Mosavi, A., ... & Varkonyi-Koczy, A. R. (2019). Performance evaluation of supervised machine learning techniques for efficient detection of emotions from online content. arXiv preprint arXiv:1908.01587.

#### [10] Thomas, B., Vinod, P., & Dhanya, K. A. (2014). Multiclass emotion extraction from sentences. International Journal of Scientific & Engineering Research, 5(2), 12-15.

#### [11] Chatterjee A, Gupta U, Chinnakotla MK, Srikanth R, Galley M, Agrawal P (2019). Understanding emotions in text using deep learning and big data. Comput Hum Behav 93:309–317

#### [12] Park, S. H., Bae, B. C., & Cheong, Y. G. (2020, February). Emotion recognition from text stories using an emotion embedding model. In 2020 IEEE international conference on big data and smart computing (BigComp) (pp. 579-583). IEEE.

#### [13] Ahmad, Z., Jindal, R., Ekbal, A., & Bhattachharyya, P. (2020). Borrow from rich cousin: transfer learning for emotion detection using cross lingual embedding. Expert Systems with Applications, 139, 112851.

#### [14] Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019.

#### [15] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

#### [16] Huang, C., Trabelsi, A., & Zaïane, O. R. (2019). Ana at semeval-2019 task 3: Contextual emotion detection in conversations through hierarchical lstms and bert. arXiv preprint arXiv:1904.00132.

#### [17] Nielsen, F. Å. (2011). A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. arXiv preprint arXiv:1103.2903.

#### [18] Cortiz, D. (2021). Exploring transformers in emotion recognition: a comparison of bert, distillbert, roberta, xlnet and electra. arXiv preprint arXiv:2104.02041.

#### [19] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020, October). Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations (pp. 38-45).

#### [20] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[21] <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>

[22] Adoma, A. F., Henry, N. M., & Chen, W. (2020, December). Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP) (pp. 117-121). IEEE.

[23] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.