

SENTIMENT ANALYSIS

Brain and Cognitive society

NANDINI BHATTAD

220693

HARDIK JINDAL

220420

Contents

1	ABOUT THE PROJECT	1
1.1	TRANSFORMERS	1
1.1.1	What are Transformers?	1
1.1.2	Brief Description Of the Transformer Architecture	1
1.1.3	Advantages of Transformers	2
1.2	WORD EMBEDDINGS	4
1.2.1	What are Word Embeddings?	4
1.3	SENTIMENT ANALYSIS	5
1.3.1	What is Sentiment Analysis?	5
1.3.2	Why do we need Sentiment Analysis?	5
2	THE MODEL	6
2.1	DESCRIPTION OF THE MODEL	6
2.2	FEATURES OF THE MODEL	7
2.3	WORKING OF THE MODEL	10
2.4	RESULTS	11

1 ABOUT THE PROJECT

1.1 TRANSFORMERS

1.1.1 What are Transformers?

Transformers are a type of deep learning model architecture introduced in the paper "Attention is All You Need". Transformers use a self-attention mechanism to process input data in parallel, making it easier to learn contextual relationships within the same sequence. This architecture has become largely used in various applications of natural languages processing, including sentiment analysis, because it captures long-range dependencies and context within data.

1.1.2 Brief Description Of the Transformer Architecture

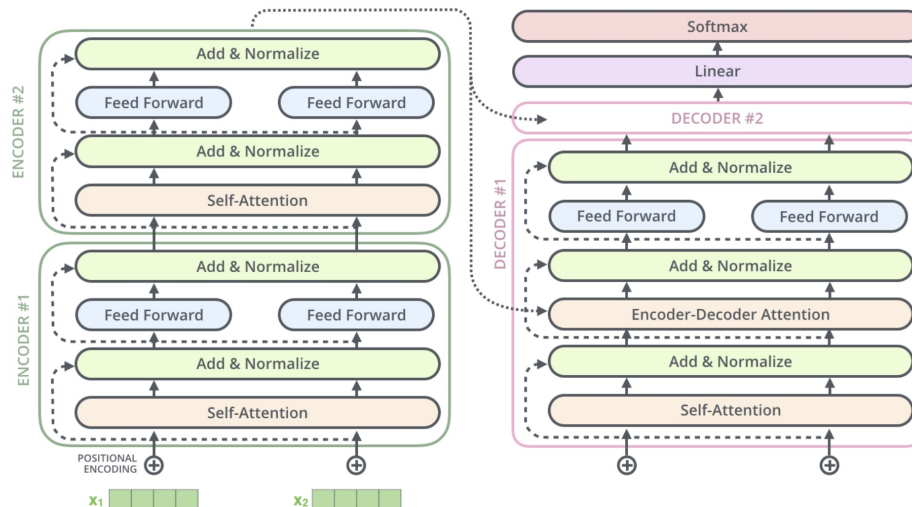


Figure 1. Transformer Architecture

- **Self-Attention Mechanism:**

The core innovation of the Transformer is the self-attention mechanism. This mechanism allows the model to weigh the importance of different words in a sequence when processing each word, capturing dependencies regardless of their positions in the sequence.

- **Multi-Head Attention:**

To enhance the expressiveness of self-attention, the Transformer uses multiple attention heads in parallel. Each head learns a different attention pattern, and their outputs are concatenated and linearly transformed.

- **Positional Encoding:**

Transformers do not inherently understand the order of elements in a sequence, so positional encoding is added to the input embeddings to provide information about the positions of tokens in the sequence.

- **Encoder and Decoder Stacks:**

The Transformer architecture consists of a stack of identical layers for both the encoder and decoder. Each layer contains a multi-head self-attention mechanism and a position-wise feedforward network. Skip connections and layer normalization are employed to stabilize training.

- **Encoder:**

The encoder processes the input sequence, such as a sentence, producing a set of context-aware representations for each token in the sequence.

- **Decoder:**

The decoder takes the output of the encoder and generates the output sequence. It also uses masked self-attention to attend to positions before the current one during the decoding process.

- **Position-wise Feedforward Networks:**

Each attention head is followed by a position-wise feedforward network, which applies a linear transformation followed by a non-linear activation function (commonly a ReLU).

- **Layer Normalization and Residual Connections:**

Each sub-layer in the encoder and decoder has layer normalization and a residual connection. These mechanisms help with the training stability of deep networks.

1.1.3 Advantages of Transformers

- **Parallelization and Efficiency:** Transformers enable parallelization of training because attention mechanisms allow each element in the sequence to be processed independently. This results in faster training times compared to traditional sequential models.

-
- **Long-Range Dependencies:** The self-attention mechanism in transformers allows the model to capture long-range dependencies in input sequences. This is particularly beneficial for tasks involving understanding context and relationships between distant elements in a sequence.
 - **Scalability:** Transformers scale well with the size of the dataset and model. Larger models, with more parameters, often lead to better performance, and transformers can efficiently handle increased model complexity.
 - **Flexibility in Sequence Length:** Unlike recurrent neural networks (RNNs) that suffer from vanishing or exploding gradients with longer sequences, transformers handle variable-length sequences effectively due to their attention mechanism, making them more versatile.
 - **Transfer Learning and Pre-training:** Transformers have been highly successful in transfer learning scenarios. Models like BERT (Bidirectional Encoder Representations from Transformers) pre-train on large corpora and then fine-tune on specific tasks, achieving state-of-the-art results in various NLP benchmarks.
 - **Global Context Understanding:** Transformers excel at capturing global context information, allowing them to understand relationships and dependencies across the entire input sequence. This is crucial for tasks like sentiment analysis, where understanding the overall context is important.
 - **Reduced Vanishing Gradient Problem:** Transformers mitigate the vanishing gradient problem associated with traditional recurrent architectures, as information can be directly propagated to any position in the sequence through the self-attention mechanism.

Overall, transformers have achieved state-of-the-art performance on various benchmarks and competitions across a range of NLP tasks, establishing themselves as the go-to architecture for many applications.

1.2 WORD EMBEDDINGS

1.2.1 What are Word Embeddings?

Word embeddings are numerical representations of words in a continuous vector space, where words with similar meanings are represented by similar vectors. These representations are learned from large amounts of textual data using machine learning algorithms, typically through techniques like Word2Vec, GloVe (Global Vectors for Word Representation), and fastText.

The basic idea behind word embeddings is to capture the semantic relationships between words based on their contexts in the data. Traditional methods like one-hot encoding represent words as sparse vectors with each word being a unique dimension. However, word embeddings aim to encode semantic information in a more compact and meaningful way.

Here are some key characteristics of word embeddings:

- **Semantic Similarity:** Words with similar meanings have similar vector representations. For example, in a good word embedding model, the vectors for "king" and "queen" might be closer together than the vectors for "king" and "car."
- **Contextual Information:** Word embeddings capture context, meaning that the representation of a word depends on its surrounding words in a sentence or document. This allows embeddings to capture different senses of a word based on its usage in different contexts.
- **Arithmetic Operations:** Word embeddings often exhibit interesting arithmetic properties. For instance, the vector representing "king" minus "man" plus "woman" might be close to the vector representing "queen."
- **Reduced Dimensionality:** Unlike one-hot encoding, word embeddings typically have a lower dimensionality, making them computationally more efficient and easier to work with in various natural language processing (NLP) tasks.

1.3 SENTIMENT ANALYSIS

1.3.1 What is Sentiment Analysis?

Sentiment analysis, also known as opinion mining, is a natural language processing (NLP) task that involves determining the sentiment or emotional tone expressed in a piece of text. The goal is to identify and classify the sentiment as positive, negative, or neutral, providing insights into the subjective opinions or attitudes of the text's author. Sentiment analysis is widely used in various applications, including social media monitoring, customer feedback analysis, product reviews, and market research.

1.3.2 Why do we need Sentiment Analysis?

Sentiment analysis is crucial for businesses and organizations to gain actionable insights from the vast amount of textual data available on customer feedback, social media, reviews, and other sources. By leveraging natural language processing techniques, sentiment analysis helps decipher the emotional tone expressed in text. This enables businesses to understand customer perceptions, monitor brand sentiment, and identify areas for improvement in products, services, or customer interactions. Analyzing sentiment in customer reviews and social media conversations provides a real-time pulse on public opinion, helping companies respond promptly to issues, capitalize on positive feedback, and shape their brand image. Additionally, sentiment analysis is valuable in political contexts, allowing researchers and policymakers to gauge public sentiment regarding political figures, policies, or issues. Beyond marketing and customer insights, sentiment analysis is used in finance to assess market sentiment from news articles and social media, aiding traders and investors in making informed decisions. It also contributes to risk management by evaluating sentiment associated with events that may impact public perception and, consequently, financial markets. In essence, sentiment analysis provides a nuanced understanding of textual data, empowering organizations to align their strategies with public sentiment, enhance customer satisfaction, and make data-driven decisions across various domains.

2 THE MODEL

2.1 DESCRIPTION OF THE MODEL

Within this model, users can choose from four distinct features, each influencing the sentiment analysis process uniquely. These features allow customization, ensuring that the analysis is tailored to specific criteria. Our model takes user-provided text for sentiment analysis and smoothly integrates with Gradio, offering a user-friendly interface. The sentiment analysis functionality is accessible through the app.

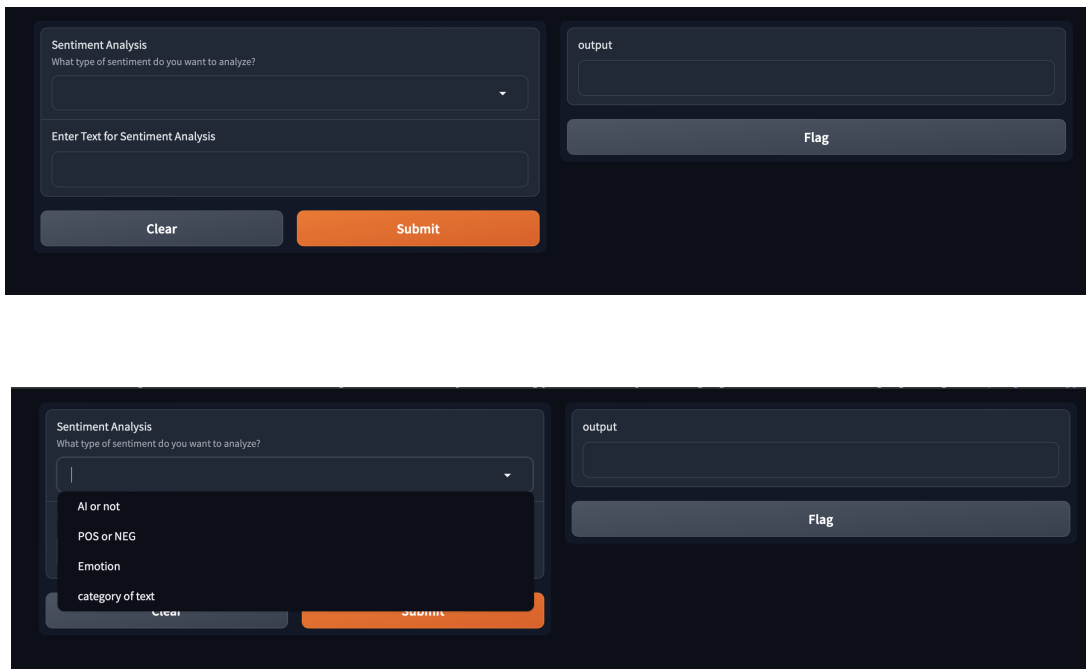


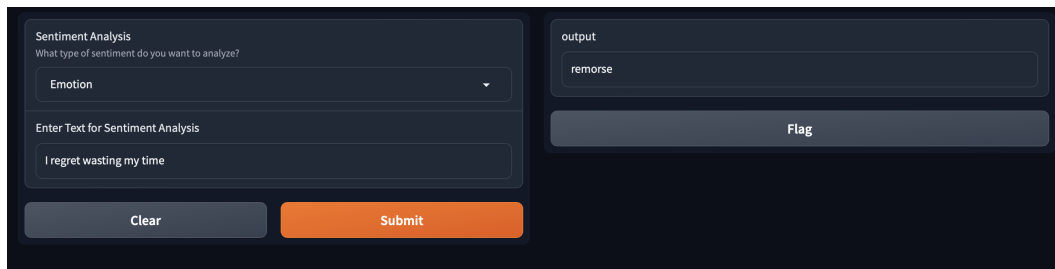
Figure 2. Gradio App Interface

The model can be accessed here:

<https://github.com/NandiniBhattad13/BYOP>

2.2 FEATURES OF THE MODEL

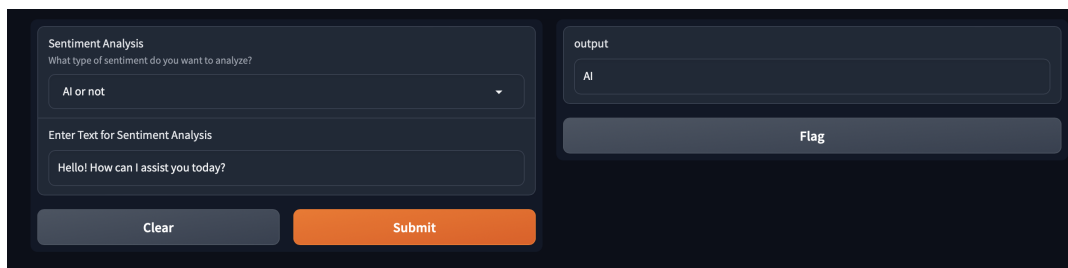
1. Emotion Analysis: The model is used to identify the emotion expressed in the seed text. The language model used was **roberta-base**, trained on the **goemotions** dataset which is based on Reddit data with around 30 emotion labels like remorse, fear, joy, grief and so on . The dataset has the 30 emotions as its columns with 0 representing false and 1 representing true. The column containing true represents the corresponding text emotion. RoBERTa simplifies and optimizes BERT's pretraining by focusing solely on the **masked language modeling(MLM)** objective. It removes the next sentence prediction(NSP) objective and trains with larger mini-batches and more training data. Emotions are often influenced by the context in which they occur. By training solely on the MLM objective, RoBERTa learns to predict masked tokens based on surrounding context. This encourages the model to capture contextual information effectively, which is crucial for understanding the emotional nuances present in text. RoBERTa generally outperforms BERT.



The screenshot shows a web interface for sentiment analysis. On the left, under 'Sentiment Analysis', there's a dropdown menu labeled 'What type of sentiment do you want to analyze?' with 'Emotion' selected. Below it is a text input field with the text 'I regret wasting my time'. At the bottom of this section are 'Clear' and 'Submit' buttons. On the right, the 'output' section displays 'remorse' and a 'Flag' button.

Figure 3. Type of Emotion

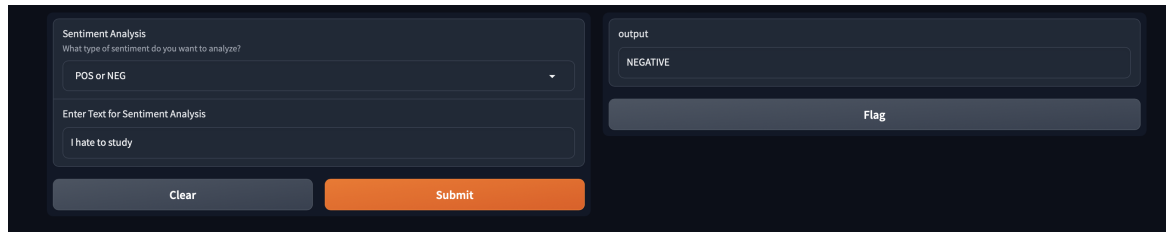
2. AI Generated Text Prediction: The model is used to recognise if the seed text is AI generated or manually written. It returns 'HUMAN' if it recognizes the text as manually written and returns 'AI' otherwise. The model is a sequence classifier based on **RoBERTa** base and then fine-tuned using the outputs of the **1.5B-parameter GPT-2 model**. The dataset consists of two columns text and label. Due to the same reasons as in Emotion Analysis, RoBERTa was used.



The screenshot shows a web interface for AI or Not prediction. On the left, under 'Sentiment Analysis', there's a dropdown menu labeled 'What type of sentiment do you want to analyze?' with 'AI or not' selected. Below it is a text input field with the text 'Hello! How can I assist you today?'. At the bottom of this section are 'Clear' and 'Submit' buttons. On the right, the 'output' section displays 'AI' and a 'Flag' button.

Figure 4. AI or Not

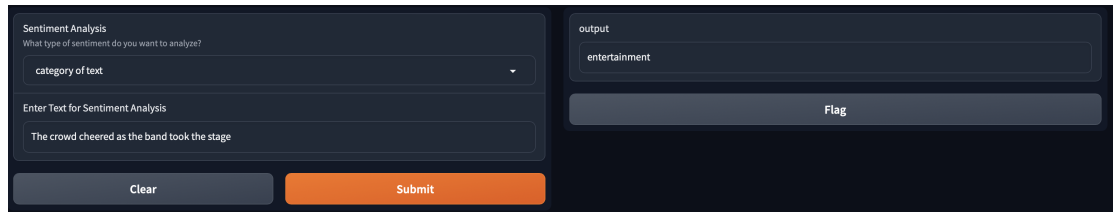
3. Positive or Negative Classification: The language model used is **DistilBERT** base uncased. The model was trained on the **imdb** dataset of movie reviews. The dataset contains two columns one of the reviews and the other containing the label 0(for negative review) and 1 (for positive review). It is used to analyse the favourability behind the text i.e. whether the sentiment is positive or negative. DistilBERT is a distilled version of BERT, meaning it is smaller and simpler. It retains much of BERT's performance while reducing the number of parameters and layers. This reduction in size leads to faster training and inference times, making DistilBERT more efficient. DistilBERT can be fine-tuned on task-specific datasets for positive and negative text classification, similar to BERT. DistilBERT's efficiency makes the fine-tuning process faster and more accessible compared to BERT, as it requires fewer computational resources.



The image shows a web application interface for sentiment analysis. On the left, under the heading "Sentiment Analysis", there is a sub-header "What type of sentiment do you want to analyze?". Below this is a dropdown menu currently showing "POS or NEG". Underneath the dropdown is a text input field labeled "Enter Text for Sentiment Analysis" which contains the text "I hate to study". At the bottom of this section are two buttons: "Clear" and "Submit". To the right of the input section is an "output" section. It contains a text box displaying the result "NEGATIVE" and a button labeled "Flag".

Figure 5. Positive or Negative

4. Category Prediction: The model is used to classify the input text in different categories like politics, environment, sports, entertainment and economy. The categories into which you want the text to be classified can be varied accordingly. The model was trained on a mixture of 33 datasets and 387 classes that have been reformatted into this universal format of five NLI datasets with around 885K texts: **"mnli", "anli", "fever", "wanli", "ling"**. Some datasets included are app_reviews, rotten_tomatoes, amazon_polarity, newsmtsc and so on. **deberta-v3-zeroshot-v1.1-all-33** proposed in the paper 'Building Efficient Universal Classifiers with Natural Language Inference' significantly outperforms the NLI-only model both on held-in and held-out tasks. Its performance on datasets it has not seen during training can expected to be around 9.4% higher than NLI-only models.



The image shows a web interface for sentiment analysis. On the left, under the heading "Sentiment Analysis", there is a question "What type of sentiment do you want to analyze?". Below this is a dropdown menu labeled "category of text" with a downward arrow. Further down is a text input field labeled "Enter Text for Sentiment Analysis" containing the text "The crowd cheered as the band took the stage". At the bottom of this section are two buttons: "Clear" and "Submit". On the right side, there is an "output" section with a box displaying "entertainment". Below this is a button labeled "Flag".

Figure 6. Category of text

2.3 WORKING OF THE MODEL

Based on the features of the model, the model performs two types of classification :

1. **Binary Classification:** The goal is to categorize instances into one of two possible classes or labels. The classes are typically denoted as positive and negative, and the task involves training a model to predict whether an instance belongs to the positive or negative class based on its features. The model employs binary classification in Positive or Negative and AI or not sentiment analysis.
2. **Multi Class Classification:** It involves distinguishing among multiple classes. Common algorithms for multi-class classification include logistic regression, decision trees, and neural networks. The model employs multi label classification in category prediction and emotion analysis.

```
imdb = load_dataset("imdb")
tokenizer1 = AutoTokenizer.from_pretrained("distilbert-base-uncased")
def preprocess_function(examples):
    return tokenizer1(examples["text"], truncation=True)
tokenized_imdb = imdb.map(preprocess_function, batched=True)
data_collator = DataCollatorWithPadding(tokenizer=tokenizer1, return_tensors="tf")
id2label = {0: "NEGATIVE", 1: "POSITIVE"}
label2id = {"NEGATIVE": 0, "POSITIVE": 1}
from transformers import create_optimizer
batch_size = 16
num_epochs = 20
batches_per_epoch = len(tokenized_imdb["train"]) // batch_size
total_train_steps = int(batches_per_epoch * num_epochs)
optimizer1, schedule = create_optimizer(init_lr=2e-5, num_warmup_steps=0, num_train_steps=total_train_steps)
model1 = TFAutoModelForSequenceClassification.from_pretrained("distilbert-base-uncased", num_labels=2, id2label=id2label, label2id=label2id)
tf_train_set = model1.prepare_tf_dataset(
    tokenized_imdb["train"],
    shuffle=True,
    batch_size=16,
    collate_fn=data_collator,)

tf_validation_set = model1.prepare_tf_dataset(tokenized_imdb["test"], shuffle=False, batch_size=16, collate_fn=data_collator,)

model1.compile(optimizer=optimizer1)
```

Figure 7. Fine tuning DistilBert for Binary Classification

2.4 RESULTS

In this section we present the performance of various models subject to the metrics of accuracy and F1 score.

- **Emotion Analysis:** It was trained on the **goemotions** dataset from the datasets library. The evaluation of the multi-label output (of the 28 dim output via a threshold of 0.5 to binarize each) using the dataset gives an accuracy of 0.474 and F1 score 0.450
- **AI or Not:** The RoBERTA base model was fine tuned on the 1.5-B parameters **GPT-2 outputs** dataset. The accuracy of the model is 0.95
- **Positive or Negative:** The model was trained on the **imdb** dataset of the datasets library. The model gives an accuracy of 0.907
- **Category Prediction:** The model was trained on a mixture of 33 datasets and 387 classes that have been reformatted into this universal format of **five NLI datasets** with around 885K texts: "mnli", "anli", "fever", "wanli", "ling". We use balanced accuracy as our main evaluation metric as many of our datasets are class imbalanced and the metric is easier to interpret than F1 macro. The balanced accuracy for the model is 0.866

Feature	Accuracy
AI or not	0.95
Emotion analysis	0.474
Positive or Negative	0.907
Category	0.866

Table 1: Accuracy of different models

References

- [1] Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF.
DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2020
- [2] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis , Luke Zettlemoyer, Veselin Stoyanov
RoBERTa: A Robustly Optimized BERT Pretraining Approach *arXiv preprint arXiv:1907.11692*, 2020
- [3] Moritz Laurer, Wouter van Atteveldt[‡] , Andreu Casas[†] , Kasper Welbers[‡] Building Efficient Universal Classifiers with Natural Language Inference *arXiv preprint arXiv:2312.17543*, 2023.
- [4] facebook/bart-large-mnli Zero-Shot Classification
<https://huggingface.co/tasks/zero-shot-classification>
- [5] Irene Solaiman, Miles Brundage, Sarah Kreps, Jasmine Wang Release Strategies and the Social Impacts of Language Models In *OpenAI Report*, 2019
- [6] Jan Hendrik Kirchner, Lama Ahmad, Scott Aaronson, Jan Leike New AI classifier for indicating AI-written text
- [7] Harika Abburi¹ , Michael Suesserman² , Nirmala Pudota¹ , Balaji Veeramani² , Edward Bowen² and Sanmitra Bhattacharya² Generative AI Text Classification using Ensemble LLM Approaches *arXiv preprint arXiv:2309.07755v1*, 2023
- [8] Alifia Ghantiwala Emotions Using NLP Based on Classifying Text <https://www.analyticsvidhya.com/blog/2022/02/analysing-emotions-using-nlp/>
- [9] Evaluating the roberta-base-goemotions model <https://github.com/samlowe/goemotions-dataset/blob/main/eval-roberta-base-goemotions.ipynb>
- [10] Barbieri, Francesco and Espinosa Anke, Luis and Camacho-Collados, Jose Multilingual Language Models in Twitter for Sentiment Analysis and Beyond. Proceedings of the Thirteenth Language Resources and Evaluation Conference
- [11] Nishanth Gadey Twitter Sentiment Analysis - XGBoost <https://www.kaggle.com/code/nishanthgadey/twitter-sentiment-analysis-xgboost>
- [12] Emotion-Classification-bert-base <https://huggingface.co/EasthShin/Emotion-Classification-bert-base/tree/main>

-
- [13] Building NLP Web Apps With Gradio And Hugging Face Transformers
https://github.com/chuachinhon/gradio_nlp
 - [14] Jay Alammar The Illustrated GPT-2 (Visualizing Transformer Language Models)
<https://jalammar.github.io/illustrated-gpt2/>
 - [15] Andrej Karpathy The spelled-out intro to language modeling: building makemore
<https://www.youtube.com/watch?v=PaCmpygFfXo>
 - [16] Jay Alammar The Illustrated Word2vec
<https://jalammar.github.io/illustrated-word2vec/>
 - [17] Gradio <https://www.gradio.app/docs/interface>
 - [18] Sentiment Analysis <https://github.com/vivekn/sentiment>