SENTIMENT ANALYSIS

Using Natural Language Processing

ABSTRACT

The increasing volume of short form textual data, such as tweets and reviews, poses a challenge for sentiment analysis. The difficulty lies in accurately identifying the sentiment expressed in the text, especially when the language used is informal and non-standard. This makes it challenging to perform automated sentiment analysis on such data. Our application intends to isolate the intent of such forms of text and classify them for further business analysis and decision making.

The objective of this project is to develop an automated sentiment analysis system for short form textual data. The system should be able to accurately identify the sentiment expressed in the text and provide a score or rating based on the sentiment. The scope of the project includes data collection, pre-processing, feature extraction, and classification. The system will be designed to handle different types of short form textual data, including tweets and reviews.

Table of Contents

INTRODUCTION

Sentiment Analysis, as the name suggests, means to identify the view or emotion behind a situation. It basically means to analyze and find the emotion or intent behind a piece of text or speech or any mode of communication.

As we humans communicate with each other in a way that we call Natural Language which is easy for us to interpret but it's much more complicated and messy if we really look into it.

Because there are billions of people and they have their own style of communicating, i.e. a lot of tiny variations are added to the language and a lot of sentiments are attached to it which is easy for us to interpret but it becomes a challenge for the machines.

This is why we need a process that makes the computers understand the Natural Language as we humans do, and this is what we call Natural Language Processing(NLP). And, as we know Sentiment Analysis is a sub-field of NLP and with the help of machine learning techniques, it tries to identify and extract the insights.

Fig-i (Sentiment Analysis)

DETAILED DESIGN

NLP (Natural Language Processing)

Natural Language Processing or NLP is a branch of Artificial Intelligence which deal with bridging the machines understanding humans in their Natural Language. Natural Language can be in form of text or sound, which are used for humans to communicate each other. NLP can enable humans to communicate to machines in a natural way.

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. There’s a good chance you’ve interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

Text Preprocessing

The term text processing refers to the automation of analyzing electronic text. This allows machine learning models to get structured information about the text to use for analysis, manipulation of the text, or to generate new text.

Text processing is one of the most common tasks used in machine learning applications such as language translation, sentiment analysis, spam filtering, and many others.

Since text processing is one of the machine learning uses that average technology consumers don’t even realize they’re using, but most people use apps daily that are using text processing behind the scenes.

Since our interactions with brands have become increasingly online and text-based, text data is one of the most important ways for companies to derive business insights. Text data can show a business how their customers search, buy, and interact with their brand, products, and competitors online. Text processing with machine learning allows enterprises to handle these large amounts of text data.

Tweet texts often consists of other user mentions, hyperlink texts, emoticons and punctuations. In order to use them for learning using a Language Model. We cannot permit those texts for training a model. So we have to clean the text data using various preprocessing and cleansing methods.

Stemming/ Lemmatization

Stemming is the process of reducing a word to its stem that affixes to suffixes and prefixes or to the roots of words known as "lemmas". Stemming is important in natural language understanding (NLU) and natural language processing (NLP).

Stemming is a part of linguistic studies in morphology as well as artificial intelligence (AI) information retrieval and extraction. Stemming and AI knowledge extract meaningful information from vast sources like big data or the internet since additional forms of a word related to a subject may need to be searched to get the best results. Stemming is also a part of queries and internet search engines.

Recognizing, searching and retrieving more forms of words returns more results. When a form of a word is recognized, it's possible to return search results that otherwise might have been missed. That additional information retrieved is why stemming is integral to search queries and information retrieval.

When a new word is found, it can present new research opportunities. Often, the best results can be attained by using the basic morphological form of the word: the lemma. To find the lemma, stemming is performed by an individual or an algorithm within an AI system. Stemming uses a number of approaches to reduce a word to its base from whatever inflected form is encountered.

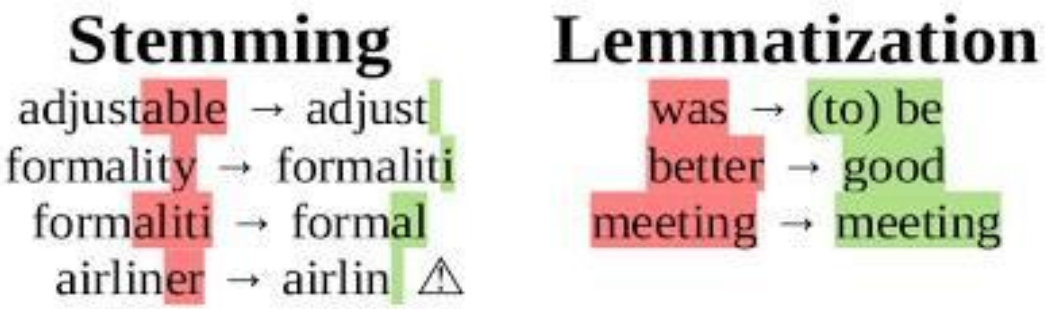

Fig- ii (Difference between Stemming & Lemmatization)

It can be simple to develop a stemming algorithm. Some simple algorithms will simply strip recognized prefixes and suffixes. However, these simple algorithms are prone to error. For example, an error can reduce words like laziness to lazi instead of lazy. Such algorithms may also have difficulty with words whose inflectional forms don't perfectly mirror the lemma, such as saw and see.

For grammatical reasons, documents are going to use different forms of a word, such as write, writing and writes. Additionally, there are families of derivationally related words with similar meanings. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

Stemming usually refers to a process that chops off the ends of words in the hope of achieving goal correctly most of the time and often includes the removal of derivational affixes.

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base and dictionary form of a word

Tokenization

Tokenization is one of the first steps in any NLP pipeline. Tokenization is nothing but spli;ng the raw text into small chunks of words or sentences, called tokens. If the text is split into words, then it's called 'Word Tokenization' and if it's split into sentences then it's called 'Sentence Tokenization'. Generally 'space' is used to perform the word tokenization and characters like 'periods, exclamation point and newline char are used for Sentence Tokeniza*on. We have to choose the appropriate method as per the task in hand. While performing the tokenization of a few characters like spaces, punctuations are ignored and will not be part of the final list of tokens.

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called *tokens* , perhaps at the same time throwing away certain characters, such as punctuation. The process is called **Tokenization.**

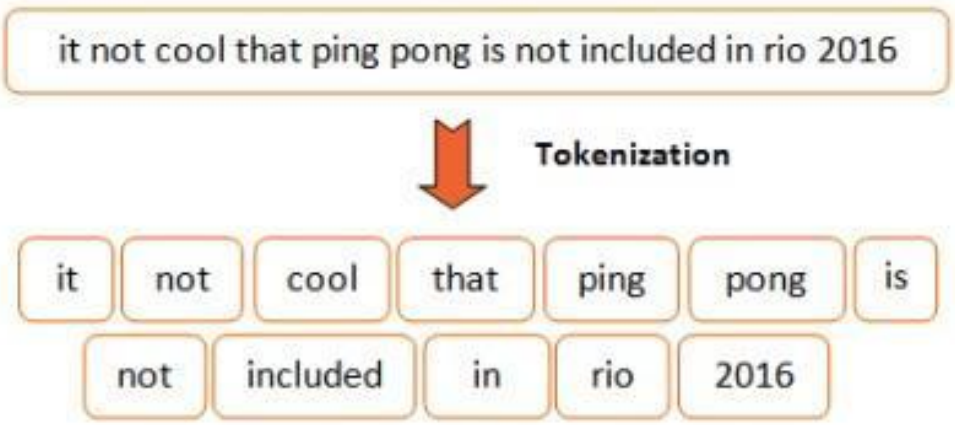

Fig-iii (Tokenization)

Word Embedding

Word embedding in NLP is an important term that is used for representing words for text analysis in the form of real-valued vectors. It is an advancement in NLP that has improved the ability of computers to understand text-based content in a better way. It is considered one of the most significant breakthroughs of deep learning for solving challenging natural language processing problems.

In this approach, words and documents are represented in the form of numeric vectors allowing similar words to have similar vector representations. The extracted features are fed into a machine learning model so as to work with text data and preserve the semantic and syntactic information. This information once received in its converted form is used by NLP algorithms that easily digest these learned representations and process textual information.

In Language Model, words are represented in a way to intend more meaning and for learning the patterns and contextual meaning behind it.

Word Embedding is one of the popular representation of document vocabulary.It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc.

Basically, it's a feature vector representation of words which are used for other natural language processing applications.

We could train the embedding ourselves but that would take a while to train and it wouldn't be effective. So going in the path of Computer Vision, here we use **Transfer Learning**. We download the pre-trained embedding and use it in our model.

The pretrained Word Embedding like **GloVe & Word2Vec** gives more insights for a word which can be used for classification.

IMPLEMENTATION

Model Training – LSTM

We are clear to build our Deep Learning model. While developing a DL model, we should keep in mind key things like Model Architecture, Hyperparameter Tuning and Performance of the model.

As you can see in the word cloud, some words are predominantly featured in both Positive and Negative tweets. This could be a problem if we are using a Machine Learning model like Naive Bayes, SVD, etc.. That's why we use Sequence Models.

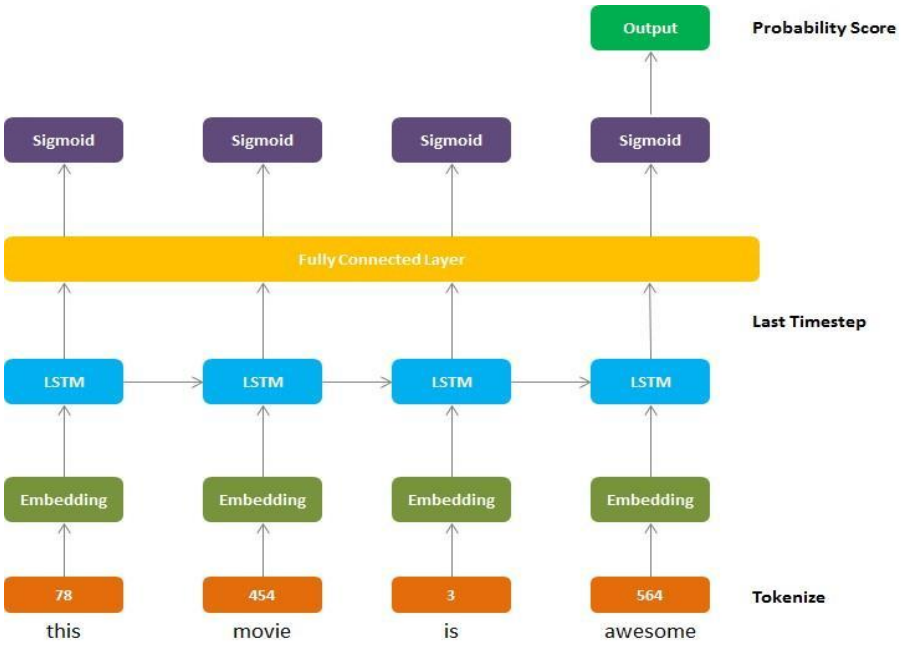

Fig-iv (Sequence Model)

Recurrent Neural Networks can handle a sequence of data and learn a pattern of input sequence to give either sequence or scalar value as output. In our case, the Neural Network outputs a scalar value prediction.

For model architecture, we use

1) **Embedding Layer** - Generates Embedding Vector for each input sequence.

2) **Conv1D Layer** - Its using to convolve data into smaller feature vectors.

3) **LSTM** - Long Short Term Memory, its a variant of RNN which has memory state cell to learn the context of words which are at further along the text to carry contextual meaning rather than just neighbouring words as in case of RNN.

4) **Dense** - Fully Connected Layers for classification

```
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedding_sequences = embedding_layer(sequence_input)
x = SpatialDropout1D(0.2)(embedding_sequences)
x = Conv1D(64, 5, activation='relu')(x)
x = Bidirectional(LSTM(64, dropout=0.2, recurrent_dropout=0.2))(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
outputs = Dense(1, activation='sigmoid')(x)
model = tf.keras.Model(sequence_input, outputs)
```

Fig-v (Implementation Of Sequence Model)

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 30)]              0

 embedding (Embedding)       (None, 30, 300)           87172500

 spatial_dropout1d (SpatialD  (None, 30, 300)          0
 ropout1D)

 conv1d (Conv1D)             (None, 26, 64)            96064

 bidirectional (Bidirectiona  (None, 128)              66048
 l)

 dense (Dense)               (None, 512)               66048

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 512)               262656

 dense_2 (Dense)             (None, 1)                 513

=================================================================
Total params: 87,663,829
Trainable params: 491,329
Non-trainable params: 87,172,500
_________________________________________________________________
```

Fig-vi (Output Of Sequence Model)

RESULT

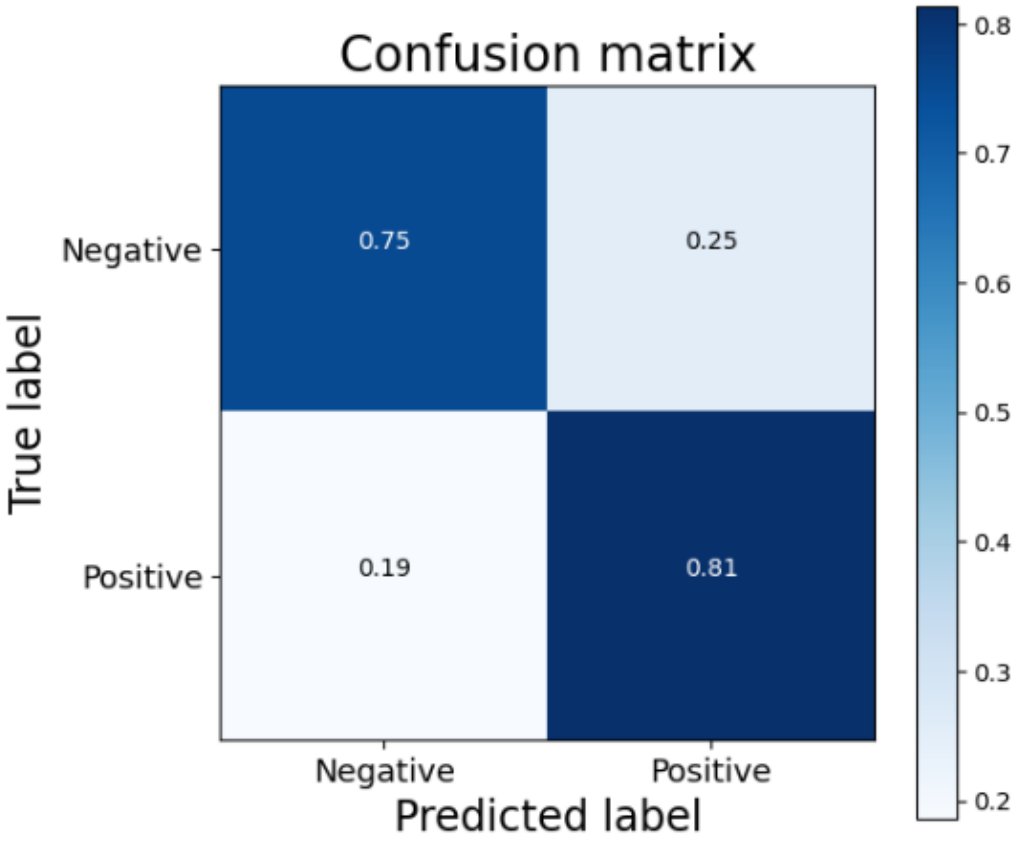

Fig-vii (Confusion Matrix)

Classification Scores

```
print(classification_report(list(test_data.sentiment), y_pred_1d))
```

```
              precision    recall  f1-score   support

    Negative       0.80      0.75      0.77    160542
    Positive       0.76      0.81      0.79    159458

    accuracy                           0.78    320000
   macro avg       0.78      0.78      0.78    320000
weighted avg       0.78      0.78      0.78    320000
```

Fig-viii (Accuracy & Classification Report)

CONCLUSION

Sentiment Analysis Model comes into play, which takes in a huge corpus of data having user reviews and finds a pattern and comes up with a conclusion based on real evidence rather than assumptions made on a small sample of data.

Sentiment analysis is an emerging research area in text mining and computational linguistics, and has attracted considerable research attention in the past few years. Future research shall explore sophisticated methods for opinion and product feature extraction, as well as new classification models that can address the ordered labels property in rating inference. Applications that utilize results from sentiment analysis is also expected to emerge in the near future

REFERENCES

[1] Montoyo, Andrés, Patricio Martínez-Barco, and Alexandra Balahur. "Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments." *Decision Support Systems* 53.4 (2012): 675-679.

[2] Srinivasa-Desikan, Bhargav. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[3]https://www.clickworker.com/customer-blog/applications-of-natural-language-processingand-nlp-datasets/#:~:text=Natural%20Language%20Processing%20(NLP)%20is,recognition%2C%20mac hine%20translation%2C%20etc.

[4] https://www.kaggle.com/code/satishgunjal/tokenization-in-nlp/notebook

[5] Nasukawa, Tetsuya, and Jeonghee Yi. "Sentiment analysis: Capturing favorability using natural language processing." *Proceedings of the 2nd international conference on Knowledge capture*. 2003.