A Project report

on

Monolingual and Cross-lingual structured sentiment analysis

Submitted By

Yogendra Singh (18BCS113)
Ritishek Yadav (18BCS079)
Lalit Palariya (18BCS047)
Vivek Rao (18BCS110)

Under the guidance of

Prof. Sunil Saumya

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
DHARWAD

1. Introduction

The importance of online reviews plays a vital role which is the key to your hotel standing on the online portal. A precious license for delighted guests, which in turn leads to greater business and increased revenue outcomes. Precise management for your brand on online portals will reassure potential customers and motivate them to opt for the hotel without a second thought in their mind. Getting the reviews classified to gain insights from it, is now an important part of the hotel business. Reviews tell the story of how the customer feels about the services which the hotel is providing. The positive reviews can also be used to promote the good efforts of the hotel just as important as to take the negative reviews into account.

Sentiment analysis helps to improve the hotel business in several ways, from preventing a shrinking reputation in the market to understanding how the guests feel about their facility. Since there are tons of reviews available through different online platforms analyzing by themselves is no longer accountable for hotel businesses. They require accurate, reliable, fast, and efficient automated systems that can provide better findings to empower business decisions.

Sentiment analysis is indeed required to automate the process of determining whether a review expresses a positive, negative, or neutral opinion about the hotel and its services. With the help of sentiment analysis, hotels can save limitless time labeling customer data such as reviews, ratings, and comments on social media platforms. Sentiment analysis is required by the hotels to monitor their brand value on online portals, and gain information from customer feedback, and in turn, apply them to improve themselves.

For calculating the polarity score, many rule-based methods are defined for sentiment analysis using Natural Language Processing (NLP) techniques such as parsing, stemming, and tokenization alongside manually constructed rules. Firstly, it is necessary to define two lists of differing word parameters (For example positive words such as decent, greatest, lovely and negative words such as worse, horrible, poor, etc.). After this a rule-based system can be feed to the lists of predefined words, the system will give the count the of positive, negative, and neutral sentiments that appears in the review, and will return a negative sentiment if it finds more negative words than positive words, and vice versa.

2. Problem Statement: To analyze and perform aspect-based sentiment analysis over reviews dataset (i.e. Hotel reviews dataset). Identifying source and target and the sentiment of given review. We also have to predict intensity of polarity.

3. Literature Summary

Most decisions in the virtual world are made after going through what influential reviewers and peers have to say about the product/service. This is the reason why the companies are now forced to see and analyze what people are talking about them on the web. From the company's perspective, the reviews and comments become very crucial. Therefore, analyzing the comments and reviews is something that an organization cannot afford to miss.

There are different methods to analyze the sentiment of the reviews let's have a look:

1. Document-level of sentiment analysis
   Opinions are usually subjective expressions that describe people's sentiments, appraisals or feelings towards an entity or an event. Many blogs or forums allow people to express their opinion in the form of reviews and comments. When opinions are expressed in the form of reviews, instead of a simple 'Yes' or 'No', identifying the actual emotions would need a subjective analysis of the words used in the review.
   In document-level of sentiment analysis, each document focuses on a single entity or event and contains opinion from a single opinion holder. The opinion here is can be classified in to two simple classes: Positive or negative (probably neutral). For example: A product review: "I bought a new phone few days ago. It is a nice phone, though it is a little big. The touch screen is good. The voice clarity is better. I simply love the phone". Considering the words or phrases used in the review (nice, good, better, love), the subjective opinion is said to be positive. The objective opinions are measured using the star or poll system, where 4 or 5 stars are positive and 1 or 2 stars are negative.

2. Sentence-level of sentiment analysis
   To have more refined view of different opinions expressed in the document about the entities, we should move to the sentence level. This level of sentiment analysis – filters out those sentences which contain no opinion and – determines whether the opinion on the entity is positive or negative.

3. Aspect based sentiment analysis
   Document level and sentence level sentiment analysis works well when they refer to a single entity. However, in many cases people talk about entities that have many aspects or attributes. They will also have different opinions about different aspects. It often happens in product review and discussion forums. For example: "I am a Nokia phone lover. I like the look of the phone. The screen is big and clear. The camera is fantastic. But there are few downsides too; the battery life is not up-to the mark and access to WhatsApp is difficult." Categorizing the positive and negatives of this review hides the valuable information about the product. Therefore, the Aspect –based sentiment analysis focuses on the recognition of all sentiment expressions within a given document and the aspects to which the opinions refer.

4. Methodology

In this section we will be discussing about various methods and techniques we have used in our project:

4.1. Dataset

This dataset has been taken from Codalab competition 2022 (SemEval-2022 Task 10: Structured Sentiment competition). This dataset contains around 1700 rows and 96 columns describing hotel reviews and its polar expression, polarity and its intensity.

4.2. Proposed Work

After reading several research articles, it became clear that Sentiment Analysis is a worthwhile endeavor. It has a wide range of uses in a variety of industries, including for the purpose of improving the customer's experience and renewing monitoring social media comments, and determining the value of a brand etc.

4.2.1. Text Preparation

Text preparation is nothing but filtering the extracted data before analysis. It includes identifying and eliminating non-textual content and content that is irrelevant to the area of study from the data.

a. Stopword removal: Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence such as “the”, “a”, “an”, “in”.

b. Stemming: Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language. In grammar, inflection is the modification of a word to express different grammatical categories such as tense, case, voice, aspect, person, number, gender, and mood. An inflection expresses one or more grammatical categories with a prefix, suffix or infix, or another internal modification such as a vowel change.

c. Lemmatization: Unlike Stemming Lemmatization reduces the inflected words properly ensuring that the root word belongs to the language and produces a dictionary meaning. In Lemmatization root word is called Lemma. A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words.

4.2.2. One hot encoding

In one hot encoding, every word (even symbols) which are part of the given text data are written in the form of vectors, constituting only of 1 and 0 . So one hot vector is a vector whose elements are only 1 and 0. Each word is written or encoded as one hot vector, with each one hot vector being unique. This allows

the word to be identified uniquely by its one hot vector and vice versa, that is no two words will have same one hot vector representation.

a. Bags of Words (BOW): A bag of words is a representation of text that describes the occurrence of words within a document. We just keep track of word counts and disregard the grammatical details and the word order. It is called a “bag” of words because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

b. TF-IDF: Term frequency-inverse document frequency is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

$$\text{tf-idf}(t, d) = \text{tf}(t, d) * \log(N/(df + 1))$$

Where,

t — term (a word in a document)

d — document (the word across a set of documents)

N — count of corpus

4.2.3. Word Embedding

A word embedding is a way of representing text where each word in the vocabulary is represented by a real valued vector in a high-dimensional space. The vectors are learned in such a way that words that have similar meanings will have similar representation in the vector space (close in the vector space). This is a more expressive representation for text than more classical methods like bag-of-words, where relationships between words or tokens are ignored, or forced in bigram and trigram approaches.

Transforming word to high dimension vector space reduces space and increases relation between word one such example is word2vec:

a. CBOW, the current word is predicted using the window of surrounding context windows. For example, if wi-1, wi-2, wi+1, wi+2 are given words or context, this model will provide wi.

b. Skip-Gram performs opposite of CBOW which implies that it predicts the given sequence or context from the word. You can reverse the example to understand it. If wi is given, this will predict the context or wi-1, wi-2, wi+1, wi+2.

4.2.4. Sentiment Labelling: Sentiment labeling involves annotating the sentiment label (i.e., positive/negative) of a word, phrase, sentence, or document.

4.3. Proposed Models

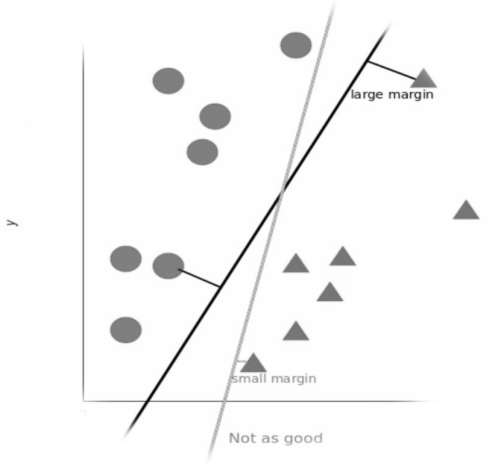
We have worked with different Machine Learning and Deep Learning models for aspect-based sentiment analysis. Machine learning algorithms can be thought of as a collection of approaches for detecting available patterns in a set of data. It makes use of previously unknown resources or patterns to predict future data (or) to put into practice. Making decisions in the face of ambiguity Machine learning has the potential to be very useful. It can be done in two ways: supervised and unsupervised. The aim is taken into account when supervised learning is done. Unsupervised learning and value (i.e. label) are carried out by not taking into account the goal value (i.e. label). There are numerous options. Types of supervised learning algorithms, such as categorizing (Decision tree, Naive Bayes etc.) and Clustering is an unsupervised learning algorithm.

4.3.1. Linear Regression: Linear regression is a statistical procedure that uses X information to predict a Y result. The data sets are analyzed using machine learning to see if there is a correlation. The relationships are then plotted on an X/Y axis with a straight line connecting them to anticipate future relationships.

The relationship between the X input (words and phrases) and the Y output is calculated using linear regression (polarity). This will identify where words and phrases lie on a polarity scale ranging from "very positive" to "extremely negative," as well as everything in between.

4.3.2. SVM: A support vector machine is another supervised machine learning model, similar to linear regression but more advanced. SVM uses algorithms to train and classify text within our sentiment polarity model, taking it a step beyond X/Y prediction.

For a simple visual explanation, we'll use two tags: red and blue, with two data features: X and Y. We'll train our classifier to output an X/Y coordinate as either red or blue.



The SVM then assigns a hyperplane that best separates the tags. In two dimensions this is simply a line (like in linear regression). Anything on one side of the line is red and anything on the other side is blue. For sentiment analysis this would be positive and negative.

4.3.3. Naïve Bayes: For sentiment analysis categorization, Naive Bayes is a relatively simple collection of probabilistic algorithms that assigns a probability that a particular word or phrase should be regarded positive or negative.

This is how Bayes' theorem works in practice. The chance of A being true if B is true is equal to the probability of B being true if A is true divided by the probability of B being true:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

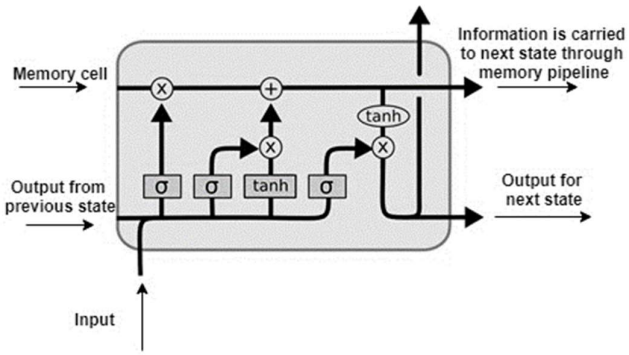
That's a lot of math, though! In a nutshell, Naive Bayes compares words to each other. We can calculate the likelihood that a word, phrase, or sentence is positive or negative using machine learning models trained for word polarity.

When techniques like lemmatization, stop word elimination, and TF-IDF are used, Naive Bayes becomes increasingly accurate in terms of prediction.

4.3.4. Random forest: Random Forest is a supervised learning technique that can be used to solve problems like regression and classification. A random forest is simply a group of trees, each of which is distinct from the others. It creates numerous decision trees and then merges them to provide an absolute and stable result, which is primarily used during training and class output.

Step 1: Load hotel review dataset and apply random forest algorithm.
Step 2: The required records were selected and the decision tree is created depending on the record.
Step 3: The decision-making process is done based on the class value.
Step 4: If the class value is less than the threshold value then it is considered as false or else it is considered as true.
Step 5: The performance of random forest algorithm on the Performance Metrics such as accuracy, precision, F-measure and recall.

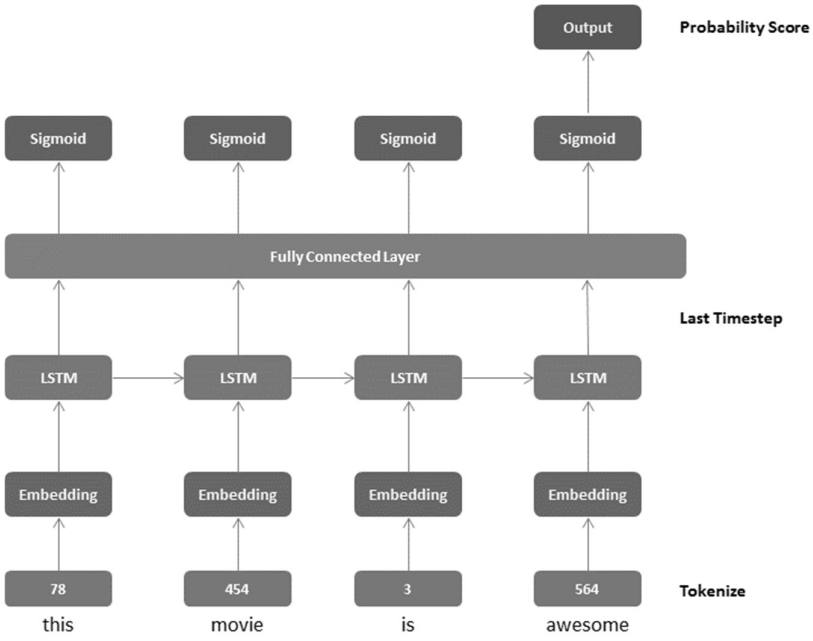
4.3.5. LSTM: It is a type of supervised deep learning algorithm. Here, the neurons are connected to themselves through time. The idea behind RNN is to remember what information was there in the previous neurons so that these neurons could pass information to themselves in the future for further analysis. It means that the information from a specific time instance (t1) is used as an input for the next time instance(t2). RNN has Vanishing gradient problem (as we go further in instance the gradient vanishes). To overcome this problem, we have LSTM (Long Short-Term Memory) LSTM is an updated version of Recurrent Neural Network to overcome the vanishing gradient problem.



It has a memory cell at the top which helps to carry the information from a particular time instance to the next time instance in an efficient manner.

Building our LSTM Model layers:

1. Tokenize: This is not a layer for LSTM network but a mandatory step of converting our words into tokens (integers). we took top 1000 repeating words and tokenized it.
2. Embedding Layer: that converts our word tokens (integers) into embedding of specific size. We have implemented 100 dimensions embedding.
3. LSTM Layer: defined by hidden state dims and number of layers. We have defined 3 hidden LSTM layers having output dimension of 100,50 and 25 respectively.
4. Fully Connected Layer: that maps output of LSTM layer to a desired output size. This is the last layer having sigmoid activation function that turns all output values in a value between 0 and 1.

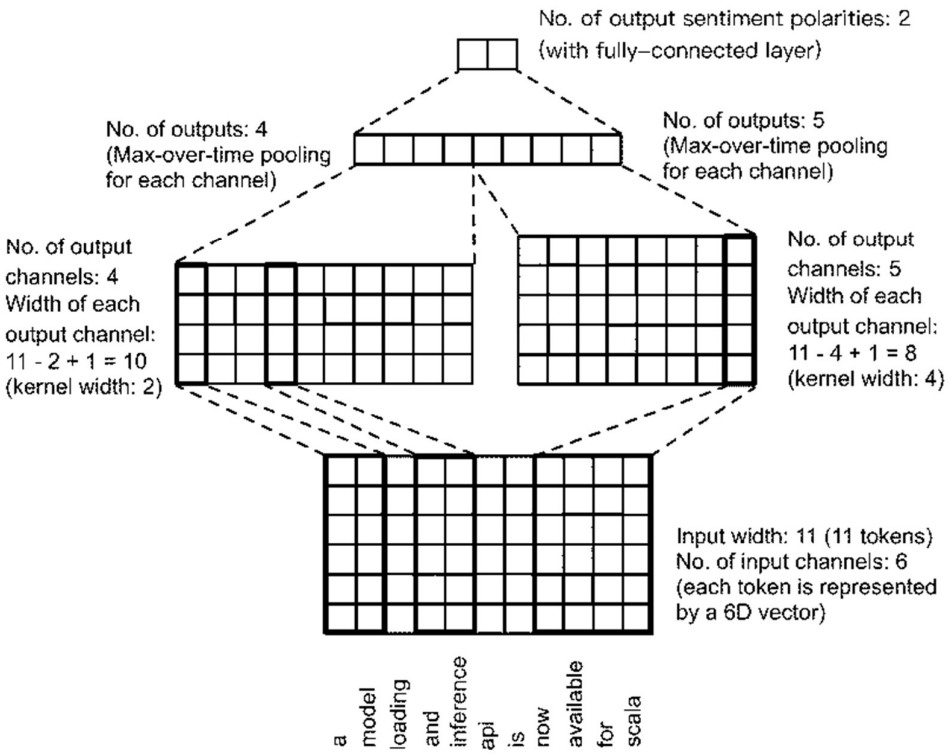


4.3.6. CNN: As the "convolutional" filter (i.e., kernel) travels over the image, CNN extracts the significant aspects of the image, which has been widely employed on image datasets. The same CNN function might be used in the text if the input data is presented as one-dimensional. Local information about texts is recorded

in the text area as the filter advances, and relevant features are extracted. As a result, utilizing CNN for text classification is a good idea.

The textCNN model accepts individual pretrained token representations as input and retrieves and alters sequence representations for the downstream application using one-dimensional convolution and max-over-time pooling. The width, height, and number of channels of the input tensor are n, 1, and d, respectively, for a single text sequence with n tokens represented by d-dimensional vectors. The following is how the textCNN model changes the input into the output:

1. Convolution operations on the inputs are performed individually using several one-dimensional convolution kernels. Local features among varied numbers of neighbouring tokens may be captured using convolution kernels of various widths.
2. Concatenate all the scalar pooling outputs as a vector after performing max-over-time pooling on all the output channels.
3. Using the fully-connected layer, convert the concatenated vector into the output categories. Overfitting can be reduced by using dropout.



4.3.7. BERT: The absence of sufficient training data is one of the most significant issues in NLP. Although there is a vast amount of text data available, we must divide it

into the many different fields in order to build task-specific datasets. And we only wind up with a few thousand or a few hundred thousand human-labeled training instances when we do this. Researchers have devised several ways for training general purpose language representation models using the massive mounds of unannotated text on the web to assist bridge this data gap (this is known as pre-training). When working with challenges like question answering and sentiment analysis, these general-purpose pre-trained models can subsequently be fine-tuned on smaller task-specific datasets.

BERT is powered by a Transformer (the attention mechanism that learns contextual relationships between words in a text). A simple Transformer consists of an encoder that reads text input and a decoder that generates a task prediction. There are two types of pre-trained versions of BERT depending on the scale of the model architecture:

BERT-Base: 12-layer, 768-hidden-nodes, 12-attention-heads, 110M parameters
BERT-Large: 24-layer, 1024-hidden-nodes, 16-attention-heads, 340M parameters

We have use BERT base model to analysis sentiment analysis for hotel reviews.

4.4. Optimizers:

a. RMSProp: **Root Mean Squared Propagation**, or **RMSProp** is an extension of gradient descent and the AdaGrad version of gradient descent that uses a decaying average of partial gradients in the adaptation of the step size for each parameter. The use of a decaying moving average allows the algorithm to forget early gradients and focus on the most recently observed partial gradients seen during the progress of the search, overcoming the limitation of AdaGrad.

Update rule for RMSProp:

$$v_t^w = \beta * v_{t-1}^w + (1-\beta)(\nabla w_t)^2$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t^w + \epsilon}} * \nabla w_t$$

$$v_t^b = \beta * v_{t-1}^b + (1-\beta)(\nabla b_t)^2$$
$$b_{t+1} = b_t - \frac{\eta}{\sqrt{v_t^b + \epsilon}} * \nabla b_t$$

b. Adam: Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the ‘gradient descent with momentum’ algorithm and the ‘RMSP’ algorithm.

Update rule for Adam:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$

4.5. Loss functions:

a. Binary crossentropy: Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value.

   Binary Cross Entropy is the negative average of the log of corrected predicted probabilities.

b. Logistic loss: This is the loss function used in (multinomial) logistic regression and extensions of it such as neural networks, defined as the negative log-likelihood of a logistic model that returns y_pred probabilities for its training data y_true. The log loss is only defined for two or more labels. For a single sample with true label $y \in \{0,1\}$ and a probability estimate $p=Pr(y=1)$, the log loss is:

$$L_{\log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

4.6. Performance metrices

a. Accuracy: Classification accuracy is perhaps the simplest metric to use and implement and is defined as the number of correct predictions divided by the total number of predictions, multiplied by 100. We can implement this by comparing ground truth and predicted values in a loop or simply utilize the scikit-learn module to do the heavy lifting for us (not so heavy in the case).

b. Precision: Precision is the ratio of true positives and total positives predicted:

$$P = \frac{TP}{TP+FP} = \frac{\text{Cancer patients correctly identified}}{\text{Cancer patients correctly identified+incorrectly labelled cancer patients as non-cancerous}}$$

$0<P<1$

The precision metric focuses on Type-I errors **(**FP). A Type-I error occurs when we reject a true null Hypothesis(H^o)**.** So, in this case, Type-I error is incorrectly labelling cancer patients as non-cancerous.

A precision score towards 1 will signify that your model didn't miss any true positives and is able to classify well between correct and incorrect labelling of cancer patients. What it cannot measure is the existence of Type-II error, which is false negatives – cases when a non-cancerous patient is identified as cancerous.

A low precision score (<0.5) means your classifier has a high number of false positives which can be an outcome of imbalanced class or untuned model hyperparameters. In an imbalanced class problem, you must prepare your data beforehand with over/under-sampling or focal loss in order to curb FP/FN.

c. Recall: It is essentially the ratio of true positives to all the positives in ground truth.

$$R = \frac{TP}{TP+FN} = \frac{\text{Cancer patients correctly identified}}{\text{Cancer patients correctly identified+incorrectly labelled non-cancer patients as cancerous}}$$

0<R<1

The recall metric focuses on type-II errors (FN). A type-II error occurs when we accept a false null hypothesis(H^o)**.** So, in this case, type-II error is incorrectly labelling non-cancerous patients as cancerous.

Recall towards 1 will signify that your model didn't miss any true positives and is able to classify well between correctly and incorrectly labelling of cancer patients. What it cannot measure is the existence of type-I error which is false positives i.e., the cases when a cancerous patient is identified as non-cancerous.

A low recall score (<0.5) means your classifier has a high number of false negatives which can be an outcome of imbalanced class or untuned model hyperparameters. In an imbalanced class problem, you must prepare your data beforehand with over/under-sampling or focal loss in order to curb FP/FN.

d. F1 score: The F1-score metric uses a combination of precision and recall. In fact, the F1 score **is the harmonic mean of the two.** The formula of the two essentially is:

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

Now, a high F1 score symbolizes a high precision as well as high recall. It presents a good balance between precision and recall and gives good results on imbalanced classification problems.

A low F1 score tells you (almost) nothing — it only tells you about performance at a threshold. Low recall means we didn't try to do well on very much of the

entire test set. Low precision means that, among the cases we identified as positive cases, we didn't get many of them right.

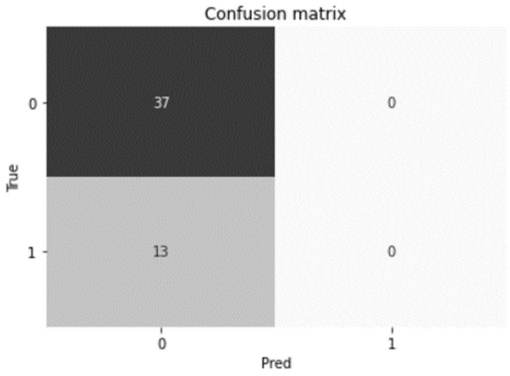
But low F1 doesn't say which cases. High F1 means we likely have high precision and recall on a large portion of the decision (which is informative). With low F1, it's unclear what the problem is (low precision or low recall?), and whether the model suffers from type-I or type-II error.

So, is F1 just a gimmick? Not really, it's widely used, and considered a fine metric to converge onto a decision, but not without some tweaks. Using FPR (false positive rates) along with F1 will help curb type-I errors, and you'll get an idea about the villain behind your low F1 score.

5. Results and Analysis

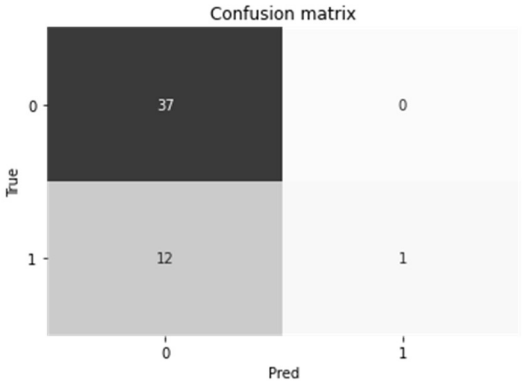
5.1.1. Logistic Regression: we had used logistic regression with L2 Normalization and we have achieved 74% accuracy. We have got around 37 true predicted label and 13 false predicted sentiments.

Confusion Matrix:



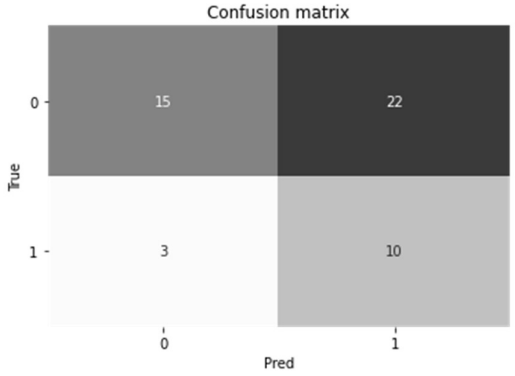
5.1.2. SGD: we had used stochastic gradient decent algorithm and have achieved around 76% accuracy. We have got around 38 true predicted label and 12 false predicted sentiments.

Confusion Matrix:



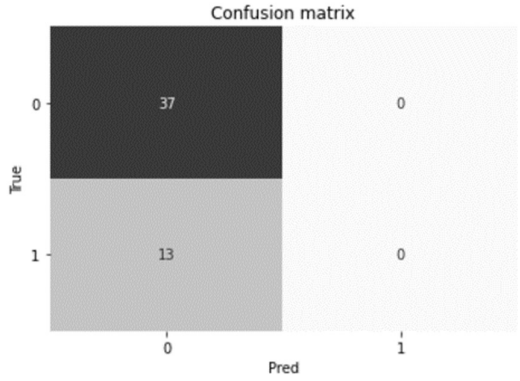
5.1.3. Naïve Bayes: we had Naïve bayes algorithm which is based on bayes therom and we have achieved around 50% accuracy. We have got around 25 true predicted label and 25 false predicted sentiments.

Confusion Matrix:

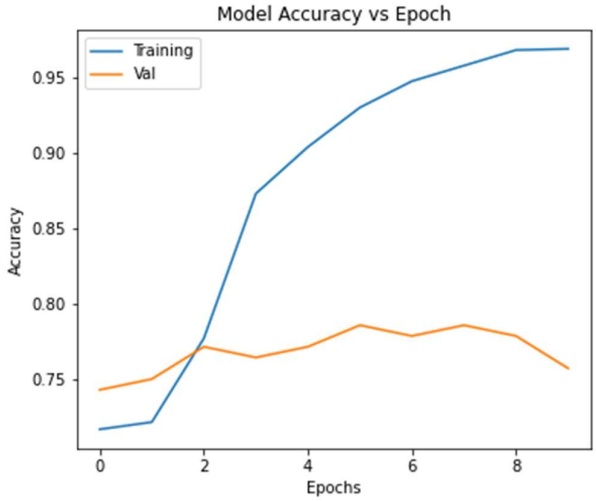


5.1.4. Random forest: we had used Random forest tree having max_depth as 2 and n_estimators: 5 as best parameters and have achieved around 74% accuracy. We have got around 37 true predicted label and 13 false predicted sentiments.

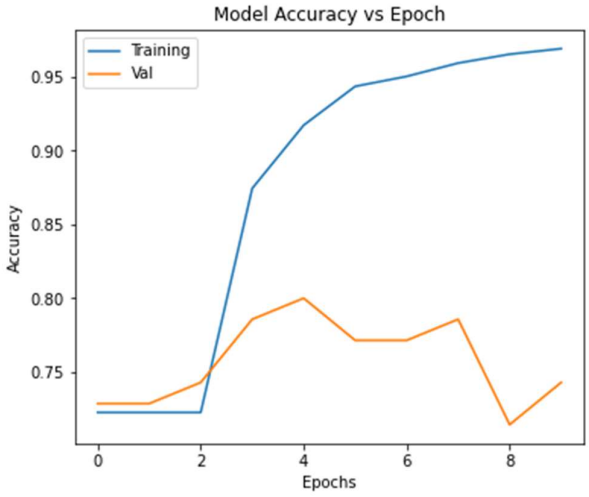
Confusion Matrix:



5.1.5. LSTM: we had implemented LSTM having 3 hidden layers with embedding dimension as 100. We tried with different optimizers and got around 77% of accuracy. After training for 2-3 epochs the validation accuracy decreases rapidly indicating the model started overfitting.



5.1.6. CNN: We have implemented CNN based deep learning architecture for our project having 2 hidden 1-D CNN layers with 100 as embedding dims. We tried with different optimizing algorithms and loss function and we have achieved around 73% accuracy. After training for 2-3 epochs the validation accuracy decreases rapidly indicating the model started overfitting.



5.1.7. BERT: We have implemented BERT base model having 12-layers, 768-hidden, 12-attention-heads, 110M parameters. After 10 epochs we have got around 99% accuracy on training dataset and 95% on validation dataset.

6. Result Comparison:

Model	Optimizers	Loss Function	Word embedding	Accuracy/F1 score
Logistic Regression		L2 norm	BOW	0.74
			TF-IDF	0.74
Naïve Bayes		GuassianNB	BOW	0.50
			TF-IDF	0.50
Random Forest			BOW	0.74
LSTM	rmsprop	Binary crossentropy		0.77
	adam	Binary crossentropy		0.79
CNN	rmsprop	Binary crossentropy		0.74
	adam	Binary crossentropy		0.73
BERT(Base)				0.95

7. Conclusion: Sentiment analysis for the hotel reviews has been carried out labeling reviews as positive sentiments which include a word like- happy, amazing, tasty, nice, pretty as well as negative sentiments which include words bad, disgusting, sad, and disappointed, etc. The whole point of the analysis is to provide suitable recommendations to the customers to select the best available option and to the business owner for successful decision making, using sentiment-based results, and implying sentiments. Moreover, the sentiment analysis in this work has been applied to determine the attitude of customers through online feedbacks given by them on hotel services, food, staff, and ambiance of the respective hotel.

8. References

- https://cs229.stanford.edu/proj2014/Vikram%20Elango,%20Govindrajan%20Narayanan,%20Sentiment%20Analysis%20for%20Hotel%20Reviews.pdf
- https://towardsdatascience.com/sentiment-analysis-for-hotel-reviews-3fa0c287d82e
- https://monkeylearn.com/blog/word-embeddings-transform-text-numbers/
- https://d2l.ai/chapter_natural-language-processing-applications/sentiment-analysis-rnn.html
- https://d2l.ai/chapter_natural-language-processing-applications/sentiment-analysis-cnn.html
- https://analyticsindiamag.com/how-to-implement-lstm-rnn-network-for-sentiment-analysis/
- https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html
- https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/