Université de Paris

UNIVERSITY OF PARIS

MACHINE LEARNING FOR DATA SCIENCE

Supevised binary classification on imbalanced Yelp restaurant dataset

Author:
Wacim Bellahcel
Imad Oualid Kacimi

Supervisor:
Louis Geiler
Mohamed Nadif

Submitted in partial fulfillment of the requirements for the Master of Machine Learning for Data Science at the University of Paris.
April 2021

Contents

1 abstract

In this Paper, we will present multiple models we have been working on for binary sentiment analysis on the yelp dataset.

We will start with a quick introduction on sentiment analysis, we will then define the problematic related and the works associated with this task.

In the second part, we will describe the dataset used, as well as the cleaning and preprocessing required to work on this kind of problems which we applied to our reviews.

Next, we will describe, analyze and compare each model used and the results obtained.

Finally, we will conclude this paper by a quick summary of our work and studies.

2 Introduction

In Data science, as the name suggests, data plays a central role, In fact every second, data is being added to Databases, and represents all the data that we have succeeded to collect.

Those data can then be used for many purposes. In this paper we will present one of the many purposes of big data, namely, sentiment analysis. Sentiment Analysis is, according to [8], "the process of identifying and recognizing or categorizing" people's emotions through a text (that come from a movie review or a commentary from a social network for example).

This type of analysis can be represented by many classes. In theory, sentiments can be divided in 2, 3 ,10 but increasing the number of classes also makes the problem harder. In this paper, sentiments are divided in two classes, 0 and 1 where 0 is a negative emotion (negative comment) and 1 is a positive emotion (positive comment).

SA is interesting due to its complexity. It's true that it's easy for someone to predict the sentiment conveyed by a text However it become much more complicated for a machine that doesn't understand human language to do so. which makes sentiment analysis a really interesting problem to work on, using our models we can predict sentiments from a text using methods that use lists of lexical features, or bidirectional encoder like BERT or even sequential models like LSTM.

All models have similarity, strengths and weaknesses that makes them a better choice depending on needs and means. As we will see in this document, each word constituting a sentence, and each sentence constituting the document, is important for the final prediction. This is why in this work we try to highlight the importance of a good document representation to increase accuracy. We will see that in more details in the next sections.

3 Problematic

Sentiment analysis is one of the natural language processing techniques for finding out the opinion or sentiment of the user regarding a product or movie or any real world thing that can have an opinion. Opinion

detection is extremely important because it helps businesses quickly understand the overall opinions of their customers and thus, marketing teams can easily use that to launch new products or to determine the versions of a product which are more popular and preferred.

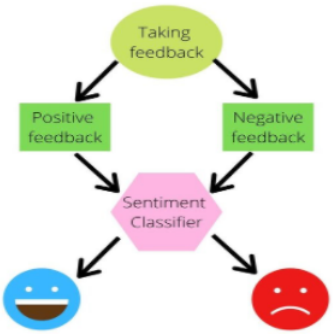

Figure 1: sentiment classification

Sentiment analysis works through natural language processing (NLP) and machine learning algorithms to automatically determine the emotional tone of online conversations. Depending on the data quantity and how accurate the model need to be, Opinion mining algorithms fall into 3 categories:

Rule-based methods which use a set of linguistic and manually-crafted rules to identify polarity, subjectivity from a document by assigning score to each word and then make an evaluation of positive and negative words, VADER completely focus on this idea. However, this approach remains naive since it doesn't take into account how words are combined in a sequence, and requires a high level fine-tuning and maintenance which can lead into very complex systems that are hard to maintain.

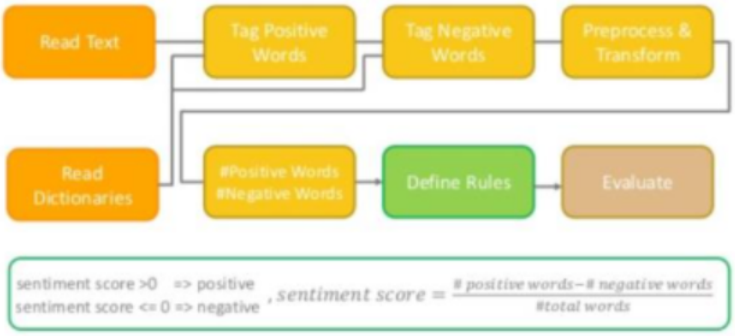

Figure 2: Rule based sentiment analysis

Automatic approaches rely on machine learning and deep learning algorithms. Actually, the sentiment analysis task is modeled as a classification problem, where a classifier receives a text and returns the category (negative or positive). The first step is to extract features from text, using classical approaches such as bag-of-word, or more efficient techniques known as word embedding which can help the model to detect similar meaning and thus improve the performance of classifiers. Then, those features are fed to an algorithm which, based on statistical techniques will learn different patterns to classify the textual data.

Figure 3: ML based sentiment analysis

Hybrid approaches, combine both rules-based approaches and automatic techniques into one system, this often leads to a more accurate model.

4 Related work

There are generally three main ways to build a model for a sentiment classification task, rule based, machine learning and deep learning methods, VADER [3], is a known rule-based method that gives decent results in sentiment classification using only manually labeled word sentiment.

In machine learning, sentiment analysis is generally considered a classification problem where different models can be applied, in [7], they explain how probabilistic and naive bayes classifiers can be used to tackle such problems.

With the booming of deep learning recently, many papers and contributions have been made in sentiment analysis and natural language processing in general in a short amount of time. As presented in [14] and [7], many deep learning architectures completely fit the needs of sentiment analysis tasks, and in 2018, a lot of concept served as a building block for deep learning in sentiment classification, like sequence to sequence models, convolutional neural network, memory networks and attention networks,

When comparing Vader or other rule-based methods to Deep learning sequence to sequence architecture like in [1] which compare LSTM trained models with VADER on sentiment classification tasks, it becomes clear that Deep learning completely outperforms rule-based methods on sentiment classification.

Thus, deep learning models became the primary focus on sentiment analysis and classification research, and many other natural language processing tasks, but building a sentiment analysis model in deep learning requires many pre-processing steps to prepare the data before training, those steps have been proved to improve the overall results.

For example in [4] they tackle the problematic of imbalanced classes and data and study the impact of imbalanced data on the overall results, it also shows how we can balance data using many resampling methods like under-sampling, over-sampling or both, the study shows that the most effective one is often under-sampling.

Also, in [5] they present several strategies of preprocessing and their impacts on classification performance. This paper explains that when we remove stop word, word without meaning, number and word with less than three characters we have a better performance on classification.

At last [10] present the benefits of pre-processing on Twitter's dataset. This article explains that when we delete stop words and noisy data the algorithm works well and when we couple that with sentiment analysis it becomes more accurate.

Thus, pre-processing is the most important step to increase accuracy of models in deep learning, but pre-processing alone isn't enough, and researcher started to think about ways to build models that fit more the

structure of the human languages and that minimize the weaknesses of recurrent neural networks like in [2], in the other hand, they found out that better words embeddings make models perform better.

Thus, more research was made on word embeddings like word2vec, or GloVe, [8] for example compares some methods for words embedding with GloVe. The most interesting comparison between GloVe and word2vec which was state of the art at the time, with CBOW. This work shows how GloVe can capture more semantics or syntactic properties of words than the others, without using neural networks.

Other research made it clear that sequence to sequence models like RNN (or LSTM / GRU) had a lot of troubles with longer text and those models were affected by a pretty known problematic in sequence deep learning models which was long term dependency (longer text made it hard for the model to "remember" earlier input) and those model were also prone to vanishing and exploding gradient.

Even though attention networks and GRU / LSTM reduced the weaknesses of seq2seq models, it didn't completely prevent that from happening, thus, other effort were made to build what we commonly call "hierarchical models", for example in [13] the paper present a "hierarchical attention network for document classification", which tries to mimic text hierarchy (word, sentence, paragraph), using this kind of hierarchy modeling, with the help attention networks, word embedding, and preprocessing, they were able to achieve state of the art result at the time.

But, even though those architecture truly reduced the long term dependency of recurrent neural network, it didn't totally vanish, adding to that the fact that "static" word embeddings were not good enough at generalizing in text classification tasks because of the fact that the context of a certain word in a sentence completely change its meaning.

In this context effort were made to try to go beyond those problematic, and in [11] Google presented their paper "Attention is all you need", which get completely rid of recurrent neural network and only use what they call "self-attention" to build a "transformer network" which is basically composed of an encoder and decoder architecture using self attention, surprisingly enough, they were able to achieve state of the art results on text translation tasks.

Which encouraged them to build and publish another paper on what they call "BERT" or Bidirectional Encoder Representations from Transformers, this model is simply the "encoder" part of their previous paper, which was trained on all the Wikipedia corpus, and, using it, they were able to achieve state of the art result on many natural language processing problems using fine tuning, an example of that is shown in [6], on this paper they use a simple BERT base network fine tuned using the SST dataset for sentiment analysis, comparing it with other well known deep learning models, they were able to show that BERT could achieve way better results in less training steps.

5 Dataset

5.1 Data description

The dataset used in this project is the Yelp restaurant reviews, Yelp is the most widely used restaurant and merchant information software across the United States. Yelp provides us with reviews and ratings provided by the users and other services offered by the restaurants such as free Wi-Fi, parking etc.

The dataset is composed of 5 Collections:
business.json This collection consists of several details related to every business such as location, business id, review count, type of food it provides, parking factors, peak hours and what type of environment it has etc

review.json The review collection of this dataset contains columns such as the user id, their rating and the message that they post as review. Natural Language Processing can be used to analyze this collection to determine what exactly the customer thinks

user.json This collection contains several details of the yelp users such as the user id, user name, since when are they using yelp, review count, friends of the users etc.

checkin.json This collection gives the day of check in and the check in time of customers. It also displays the business id corresponding to it.

tip.json The tip collection consists of a useful message from the customer to other customers. It consists of the user id of the person who posted the message and the date on which the tip was posted.

photo.json photo data including captions and labels.

In our project, we will primarily use both business.json and review.json, the first one will be used for restaurant visualisation through interactive maps and the second collection will be used to retrieve the reviews for the sentiment analysis process.

5.2 Data preparation

The initial yelp dataset is far too large, and for computational reasons, we decided to filter the data and keep only the data for the city of Richmond. Also, the original dataset contains ratings between 1 and 5 stars and since we are only interested in positive and negative sentiment, we grouped the 1-2 ratings into negative reviews, the 4-5 into positive reviews and eliminated the ones equal to 3 which represent the neutral reviews.

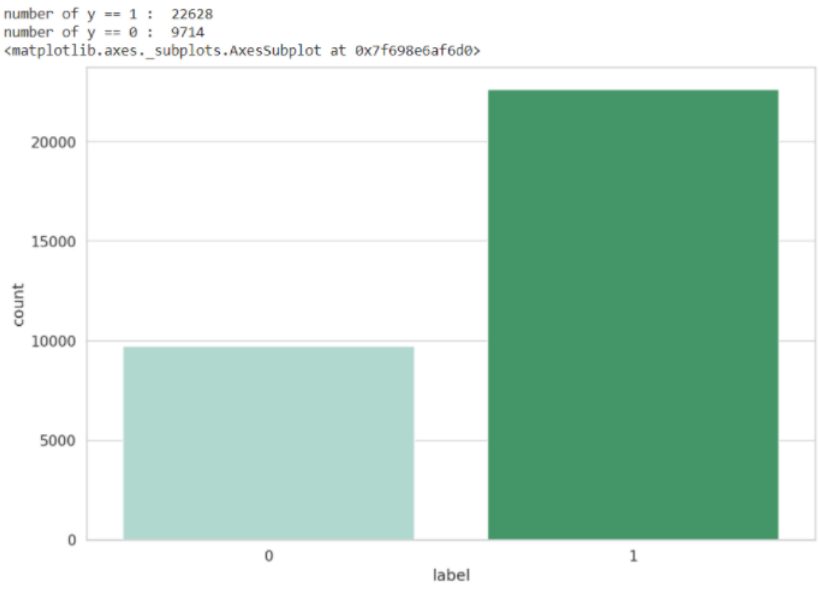

Figure 4: Yelp reviews sentiment distribution

Text preprocessing is an important step for natural language processing tasks. It is needed for transforming textual data from human language to machine-readable format for further processing. In our text preprocessing we've done the following tasks:

- **Normalization:** Refers to a series of tasks that aims for putting words from our tweets into equal footing and allows processing to proceed uniformly. It also aims to remove irrelevant information. In our case, we performed the following tasks:

  - removing URLs.
  - removing punctuation.
  - removing numbers.
  - removing non ascii characters
  - converting all text to the same case (lower case).
  - removing punctuation.

- **Tokenization:** It is the process of splitting a given text into smaller pieces called tokens. Words, numbers, punctuation marks, and others can be considered as tokens.
- **Stop words removal:** Stop words are the most common words in a language like ("the", "a", "on", "is", "all") for English language. These words do not carry any important information and are usually removed from texts, to save computing time and efforts in processing large volumes of text.
- **Lemmatization:** Lemmatization to reduce inflectional forms to a common base form.

In our project, we applied different levels of text cleaning depending on the model that will be used. For example, we used all of the previously mentioned steps when cleaning the data for the LR, SVM, and lstm models, while we only removed URLs, numbers, and no ascii characters when preparing the input data for the bert model.

5.3 Data transformation

After preprocessing our reviews, we need to transform them into numerical values so we can feed them into our logistic regression and svm models. In order to do that, we use the Doc2Vec method, which is an extension of Word2Vec but applied to a whole document (i.e. multiple reviews) instead of only one. Compared to more traditional methods such as bag of words, Doc2Vec encodes the documents in a way that each word keeps its meaning, and similarities between words will be captured, so in our case, words that represent a negative opinion would be closer (in terms of coordinates) to each other than words that are positive.

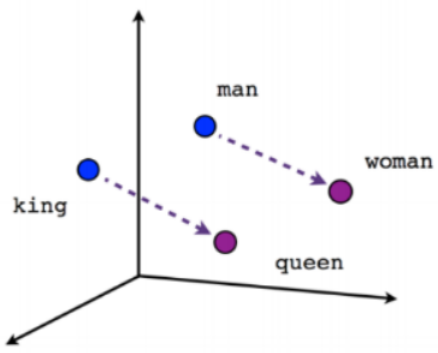

Figure 5: word 2d vectors representations

5.4 Resampling methods

An imbalanced classification problem is an example of a classification problem where the distribution of examples across the known classes is biased or skewed. The distribution can vary from a slight bias to a severe imbalance where there is one example in the minority class for hundreds, thousands, or millions of examples in the majority class or classes. Building application and optimization of learning algorithms on imbalance datasets can and will lead to semantically incorrect models.

Several different techniques exist in the practice for dealing with imbalanced dataset. In this report will highlight some of the important resampling methods that can be used.

In our implementation we choose to use the hybrid method by fusing the random over and under sampling.

Oversampling methods: Random oversampling is a naïve approach that consists on duplicating examples from the minority class, the duplication is done by randomly selecting samples from the rare class. As shown in the below Figure 29 we can see that the plot didn't change, this is due to the fact that we are only duplicating the original samples and thus the points are superposed on each other.

Several other over samplings exist such as SMOTE (Synthetic Minority Oversampling Technique), ADASYN (Adaptive Synthetic) but as mentioned before we will focus on mixing random and over and under sampling.

Under sampling methods: Random under sampling is another naive method that aims to reduce samples from the majority class by randomly selecting examples from the majority class and deleting them from the training dataset. Doing this can result into a loss in valuable information since and thus bias our model.

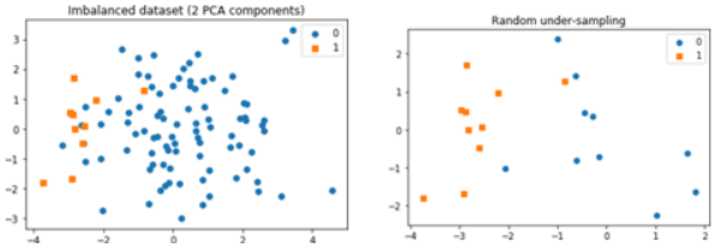

Figure 6: Random undersampling

Just like the oversampling-methods, several under sampling methods exists for example we have Tomek links but we will just focus on the random under sampling method.

Hybrid methods: Hybrid methods aim to reduce the negative effect of applying oversampling and under-sampling methods in isolation, which are: overfitting and information loss.

We can choose to fuse several over and under sampling methods. In this project we choose to combine between random over and random under sampling. To do so we first start by applying a modest amount of oversampling to the minority class to improve the bias towards these examples, whilst also applying a modest amount of undersampling to the majority class to reduce the bias on that class.

5.5 Data visualization

Multiple visualization methods were experimented to further explore the dataset, using our co-occurrence table after cleaning we first implemented correspondence analysis on our data to try to find relation between words in our vocabulary.

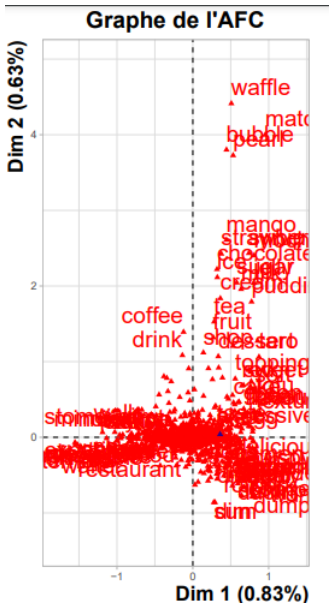

Figure 7: Dataset CA visualization

In the figure below, we can see that even thought the CA visualization isn't really useful to separate words and reviews with different sentiment polarity, it can help us understand the content of our reviews better, as we can see her, the first dimension separate between aspects of the restaurant and aspects of the food served, the second dimension in the other hand seems to separate between different kinds of food and adjectives that describe them.

The visualization below, can also show the localization of the different restaurants colored by their mean reviews, as can be seen on the map, most restaurants in the city of Richmond are located in Golden Village.

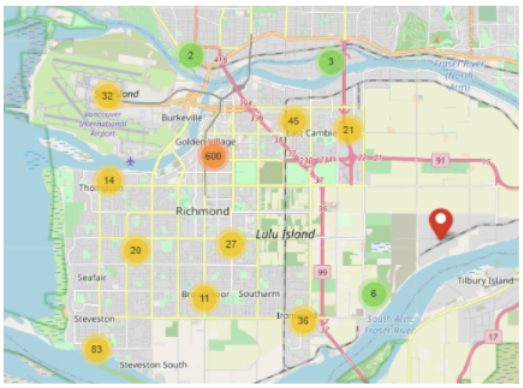

Figure 8: Geo-mapped restaurant sentiments

If we further zoom in on the map, we can clearly see that there doesn't seems to be a link between restaurant location and the overall review:

Figure 9: Zoomed geo-mapped restaurant sentiments

6 Solutions

6.1 Logistic regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. It is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

The idea of linear regression is that the data are fed into a linear regression model, which is then acted upon by a logistic function that squeezes the result between 0 and 1.
The logistic equation is:

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$

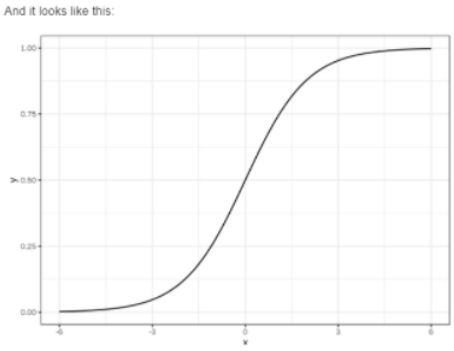

Figure 10: Logistic regression

This sigmoid function will take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO. For example: If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that the review is positive.

Model configuration and training: To find the best parameters to use for our Logistic regression classifier, we applied a simple random search.

6.2 Support Vector Machine

A support vector machine is a machine learning algorithm that learns to categorize text into two groups, it is trained in a supervised setup where you have to give it labels of each category to learn from.

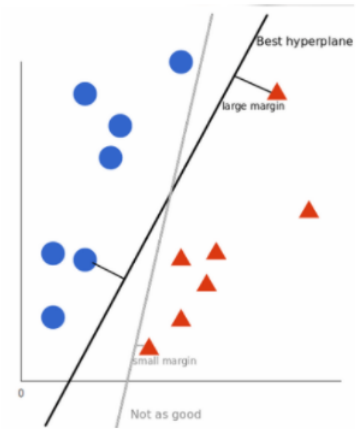

Figure 11: SVM with linear hyper plane

The idea of a support vector machine is to output a hyper plane that will best separate our data points from different classes, in a 2 dimensions setup where our features are simple 2 dimensions vectors, this would mean separating our two classes by a line that best optimize the margin between our two classes and the center separating line.

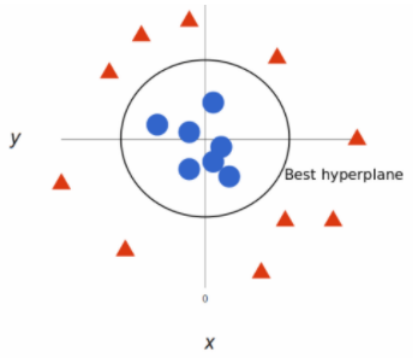

Figure 12: SVM with non-linear hyper plane

The problem becomes more complicated one we introduce non linearity, in the case where our data points can't be separated by a simple line, SVM models works by "adding" dimensions to our features points which will project them in a dimension where our two classes will become linearly separable.

To do that using SVM, we don't actually project each data point (which could become computationally expensive), another method that is used in svm is what we commonly call a kernel trick, which is simply applying a "kernel" to our calculation (a dot product in the projected dimension) to our data without projecting them in the new space.

Model Configuration and training: To find the best parameters to use for our SVM classifier, we applied a simple random search.

6.3 Long Short Term Memory network

Long short-term memory is an artificial recurrent neural network (RNN) architecture] used in the field of deep learning.

Standard recurrent neural networks suffer from short-term memory[9]. As we said earlier, if a sequence is long enough, they will have trouble transmitting information from previous stages to subsequent stages, and thus, during back propagation, standard RNNs will suffer from the vanishing gradient problem (the gradient values used to update neural network weights shrink and become extremely small, from which layers that get these small values stop learning.)

The LSTM model was created to address the problem of short memory. This is due to the internal mechanism called gates that can adjust the information flow. To better understand LSTMs, let's start by explaining the standard mechanism of standard recurrent neural networks. An RNN works as follows: words are first transformed into machine-readable vectors. Then, the network processes the sequence of vectors one by one.

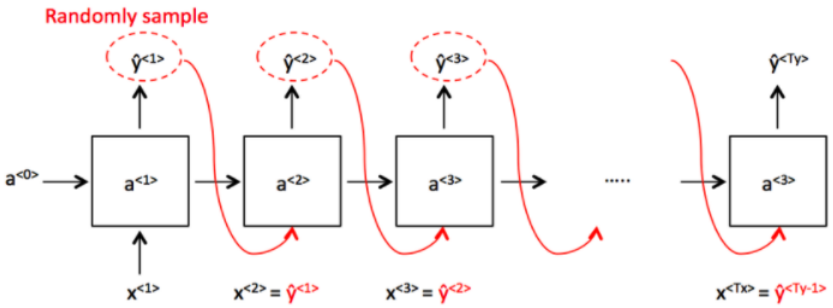

Figure 13: RNN architecture

While processing, the network passes the previous hidden state to the next step of the sequence. the hidden state acts as the neural memory and holds information on previous seen data[9]. The hidden state is calculated as follows, the input and previous hidden state are combined to form vectors and thus the resultant vector will contain information on the current and previous input, then the vector goes through the tanh activation which will output the new hidden state (memory of the network). The tanh function will ensure that values stay between -1 and 1, thus regulating the output of the neural network and preventing values from exploding.

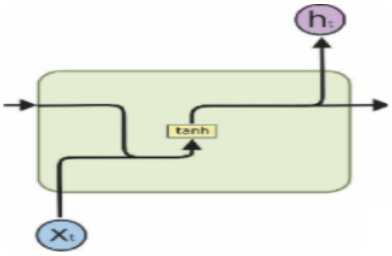

Figure 14: Basic RNN cell architechture

LSTM's networks have a similar workflow as standard RNN, the differences are the operations within the LSTM's cells, more gates are added to ensure keeping or forgetting information while propagating through the networks.

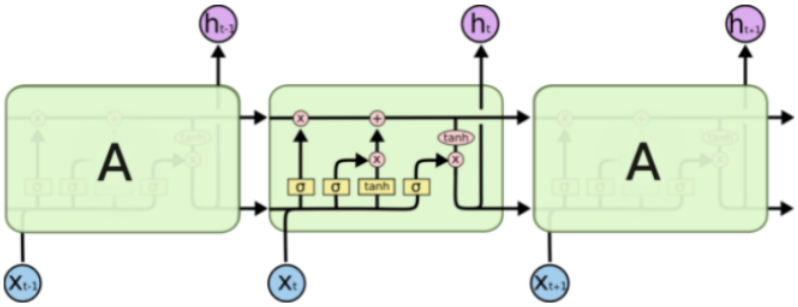

Figure 15: LSTM general and cell architecture

The core concept of LSTM is the cell state, and it's various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain. You can think of it as the "memory" of the network. The cell state, in theory, can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make it's way to later time steps, reducing the effects of short-term memory[9].

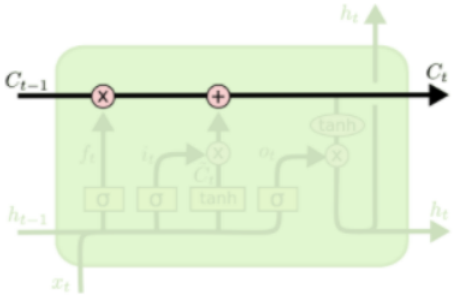

Figure 16: LSTM cell state

Step-by-Step LSTM Walk Through: The first step in our lstm is to to decide what information we're going to throw away from the cell state, this decision is made by the sigmoid layer called the "forget gate layer", it uses the previous hidden state and the new input to generate a vector of values between 0 and1, 0 when the component of the input is deemed irrelevant and closer to 1 when relevant. These output values are then sent up and point wise multiplied with the previous cell state, this will help decide which pieces of the long-term memory should be forgotten[9][12].

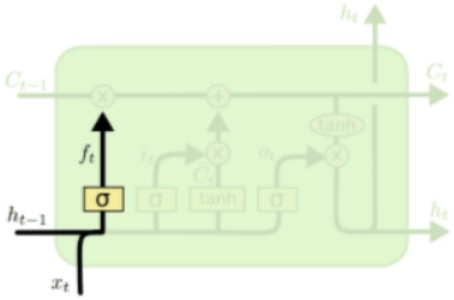

Figure 17: LSTM forget gate layer

The next step involves both new memory network and the input gate and the goal is to decide what new information we're going to store in the cell state given the previous hidden state and new input data. The new memory network is a tanh activated neural network which has learned how to combine the previous

hidden state and new input data to generate a new memory update vector which tells us how much to update each component of the long-term memory (cell state). However this step doesn't check if the new input data is worth remembering and this is where the input gare comes. It uses the sigmoid function which acts as a filter identifying which components of the new memory vector are worth retaining. The output of both tanh and sigmoid function are pointwise multiplied and then added to cell state, resulting in the long-term memory of the network being updated[9][12].

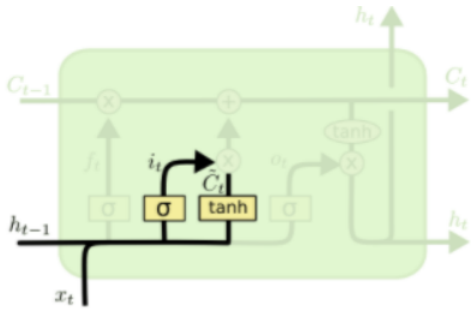

Figure 18: LSTM new memory network and input gate

The final step after updating the cell state, is the output gate which decides the new hidden states. This step uses the newly updated cell state, the previous hidden state and the new input data. we start by applying the tanh function to the current cell state to obtain the squished cell state, which now lies in [-1,1]. We then pass the previous hidden state and current input through the sigmoid function to obtain the filter vector and finally apply this filter to the cell state by pointwise multiplication to output the new hidden state[9][12].

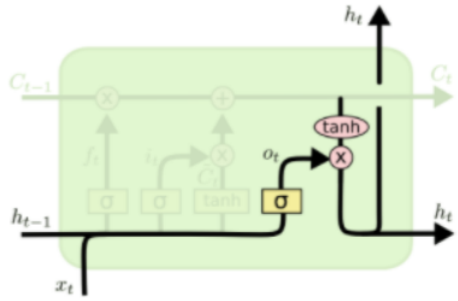

Figure 19: LSTM output gate

To review, the Forget gate decides what is relevant to keep from prior steps. The input gate decides what information is relevant to add from the current step. The output gate determines what the next hidden state should be.

Model Configuration and training : In our case, the model we trained is composed of:

- An embedding layer which will encode each input word.
- An LSTM layer followed by a dense layer.
- The output of the dense layer will be first fed to a relu activation, followed by a dropout and finally a sigmoid function will be applied to predict the sentiment.

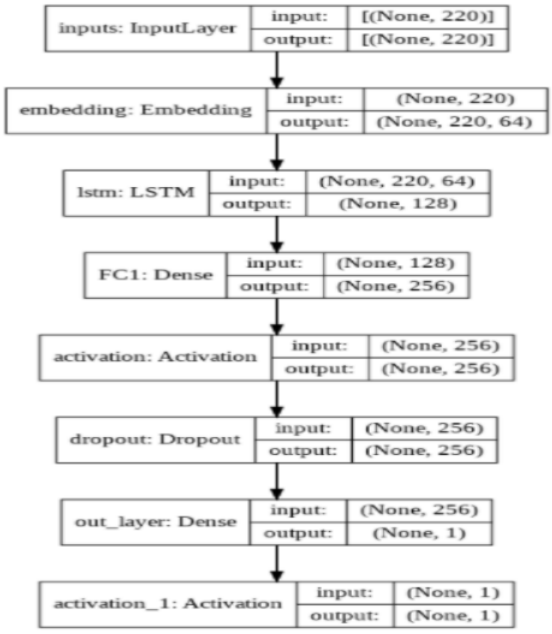

Figure 20: LSTM model architecture

6.4 BERT

"Bidirectional Encoder Representations from Transformers (BERT) is a technique for NLP (Natural Language Processing) pre-training developed by Google."

To truly understand BERT, we first have to understand Attention, how it works and why it is important.

In natural language processing, attention appeared as a way to improve sequence to sequence models performance (like recurrent neural networks) by training weights such that some words will be given more attention than others.

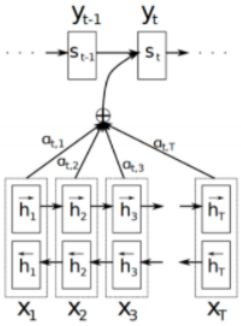

Figure 21: Attention mechanism architecture

But even if adding attention to seq2seq model increased the overall accuracy, it couldn't fixe all weaknesses related to recurrent neural networks like long term dependency or slow calculation.

In this context, researches were made to put forward the idea of transformers networks, the most popular one being "Attention is all you need" [11] and the principle of self-attention.

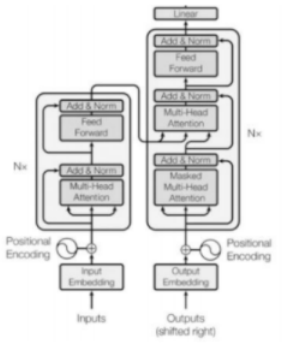

Figure 22: Encoder-Decoder from "attention is all you need"

The idea behind the transformer's architecture is to completely get rid of the sequence models (RNN or LSTM) and only keep the attention part, the attention used in a transformer architecture is commonly known as multi-head self-attention.

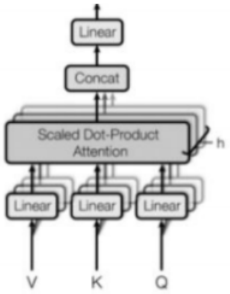

Figure 23: Multi headed self-attention

To calculate self-attention, we first need to calculate the query, key, and value vector of each input, we do that using three weight matrices that are trained during the training phase called the query, key and value matrices, we simply multiply each input by each matrices much like a fully connected layer.

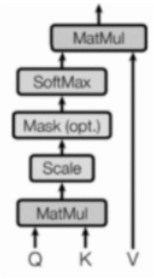

Figure 24: self-attention calculation

We can then calculate the scores for a specific input word embedding by multiplying the query by the key vector, we then use a SoftMax function and multiply the resulting scores by the value vector to get the final output.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

Using the same idea, we can stack multiple self attention networks together, forming a multi-head self-attention network.
to construct an encoder cell, we simply stack the selfattention with a feed forward network, the final transformer architecture is simply multiple stacked encoder one above the other the result is then fed to stacked decoders.

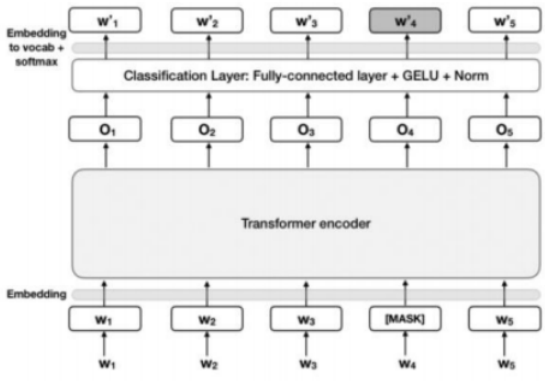

Figure 25: Bert architecture

BERT itself is completely based on a transformer architecture, in a BERT architecture, we simply get rid of the decoder part of a transformer and just keep the encoder, feeding the result to a classification layer. There are many advantages to using a BERT classifier:

- Bidirectional: Bert is naturally bidirectional.
- Generalizable: Pre-trained BERT model can be fine-tuned easily for downstream NLP tasks.
- High-Performance: Fine-tuned BERT models beat state of the art results.
- Universal: Trained on Wikipedia articles.

Model Configuration and training : When implementing a BERT classifier, we don't want to train it from scratch and BERT are built in such a way that makes it easy to use fine tuning.

Fine-tuning consists of taking a pre-trained model, in our case BERT, add an untrained layer to the end, and train the new model for our specific task (in our case sentiment classification).

There are many advantages to fine tuning:

- Quick development, since we only need to retrain the end layers, the training step is much quicker.
- Less Data, since all weights are already trained on a gigantic corpus of text.
- Often give better results.

There isn't only one BERT model published, in fact, many Bert models are available depending on number of attention heads and encoder layers, but also case sensitivity, for example:

- Bert-Base, Uncased: 12-layers, 12-heads.
- Bert-Large Uncased: 24-layers, 16-heads.

For our experiment, we worked with a BERT-Base Uncased pretrained model that we fine-tuned using our pre-processed data.

7 Results and analysis

For our results, we will primarily track and compare the performance of our models based on the F1-score and accuracy metric on the test set to make an overall result comparison of each method.

7.1 Models comparison

	Undersampling			Oversampling			Hybrid		
Model	Acc	F1	AUC	Acc	F1	AUC	Acc	F1	AUC
Logistic regression	0.84	0.82	0.834	0.84	0.82	0.8377	0.84	0.82	0.8361
SVM	0.84	0.82	0.8356	0.84	0.82	0.8374	0.84	0.82	0.8369
LSTM	0.88	0.86	0.970	0.92	0.90	0.964	0.92	0.90	0.966
BERT	0.96	0.95	0.958	0.96	0.95	0.956	0.96	0.95	0.9547

Table 1: models results on Yelp Restaurant reviews in Richmond with oversampling, undersampling and hybrid resampling methods

First thing we can notice from the table is the fact that the sampling method has no impact on the results on all our models, this is primarily due to the fact that even though there is an imbalance between the positive and negative class (the positive class representing approximately two third of the dataset), this class imbalance isn't pronounced enough to have a significant impact on training.

If we now compare our models, first comparing the different methodology used, we can see that deep learning models that take into account the order of our words in each document (LSTM being a sequence model, and bert using positional encoding) outperform traditional machine learning models (logistic regression and SVM). Nevertheless, traditional methods were still able to get decent results using word embeddings.

Now comparing LSTM and BERT, we can see that BERT is able to get significantly better results, this is due to the many benefit that comes with the use of transformers:

- BERT uses contextual word embeddings that take into account the context of a word in the sentence, Context plays a major role in prediction, what we mean by context is the fact that a word can have completely different meaning depending on the sentence where it is used.
- Text pre-processing for LSTM can lead to information and the resulting text can lose its original meaning, BERT based models don't need aggressive text preprocessing.
- Even though LSTM tries to reduce the vanishing gradient and information loss problem compared to RNN, it is still prone to this kind of problems (loss of information through multiple computation steps through gates). BERT architecture on the other hand is primarily based on a self-attention mechanism which makes it robust to this kind of issues (similarly to a fully connected layer, in a Bert model, information concerning a word is accessible at each step and not in sequence).

7.2 Results visualization

To explain and visualise our results we used Shap which is an open source tool that allows us to better explain our model outputs.

As a way to try and explain our model predictions, we visualize 4 kind of outputs ,namely true positive, true negative, false positive and false negative examples, we plot each output using SHAP force plot and text plot.

The force plot is designed to provide an overview of how other parts of the same text are combined to produce the model output prediction. positive red features are "pushing" the model output higher while

negative blue features “push” the model output lower.

Moreover, the text plot gives us the importance of each token overlayed on the original text that corresponds to that token. Red regions correspond to parts of the text that increase the output of the model when they are included, while blue regions decrease the output of the model when they are included

If we now look at each example:

- **True positive :** As we can see on the figure below, our model predicted the later sentence correctly (positive), from the token we can see that the sentence contains only positively interpreted tokens, with the tokens “I like this store” and “I will come again” having a stronger impact on the overall result pushing the model further to the positive side.

Figure 26: True positive example visualization

- **True negative :** Now looking at a true negative review, we can clearly see what pushed the model to give such output, as it interpreted most of the token to have a negative impact on the overall result, if we look closer to the token with a stronger influence, we can clearly see that the word “but” played a huge role in the final prediction, moreover the final sentence “and it seems she doesn’t. . . restaurant” play a decisive role in shifting the model output toward a negative sentiment.

Figure 27: True negative example visualization

- **False negative:** if we now focus on wrong predictions, first looking at false negatives, we can see in the examples below false negative reviews are often due to neutral reviews that contain some positive and negative aspects.

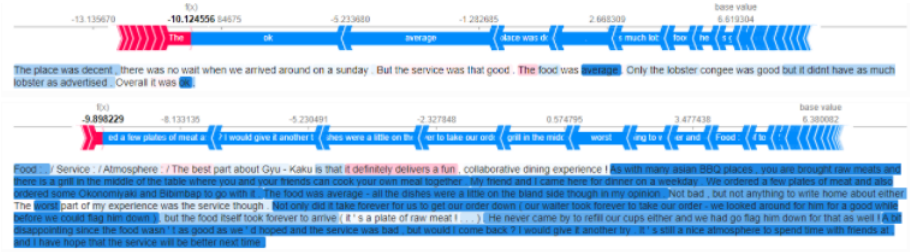

Figure 28: False negative example visualization

- **False positive :** The same observation can be made for false positive predictions, with errors due to neutral reviews, moreover, another important problem that can be highlighted using the example below are that the model can also wrongly attribute a positive label to a review containing sarcasm, in this example, even though the model seems to be able to push its output pretty close to the negative side, the token “love” used in this sentence sarcastically completely shifted the prediction toward the positive side.

Figure 29: False positive example visualization

Finally, if we now look at the average token importance in our sentences we can see that tokens which inspire a positive sentiment like “finest” or “thoughtfully” clearly have a strong positive impact, but there are also tokens that alone won’t be of any use for a sentiment classification task like “rather”, but do have a positive impact due to its use in the context of restaurant reviews.

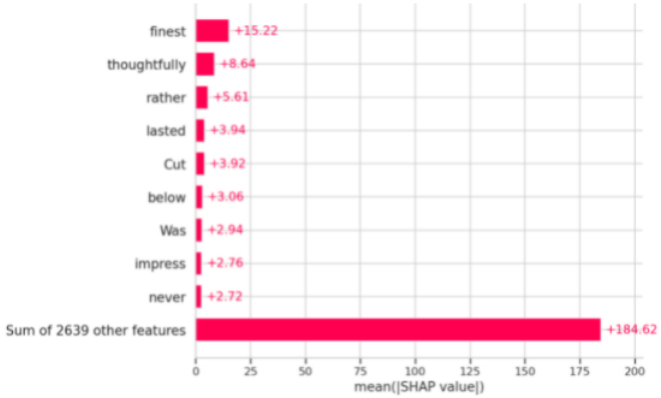

Figure 30: Average token importance

if we now plot single words importance per sentence, for example for the word “but” below, we can see that its impact clearly depends on the context and its use in the sentence.

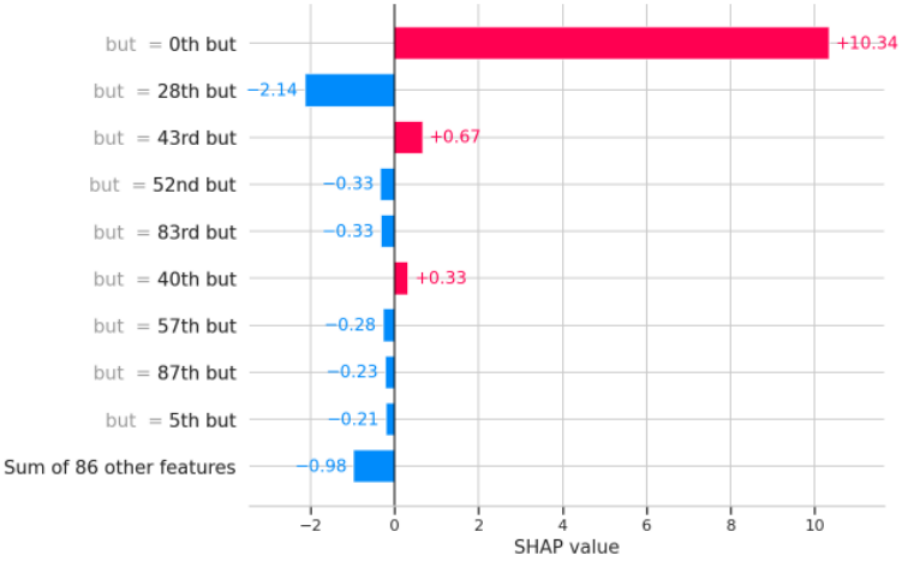

Figure 31: Importance of the word ”but”

The word “quality” on the other hand, seems to be more often used in a negative context even though it can also be used to express a positive sentiment depending on its use in the sentence.

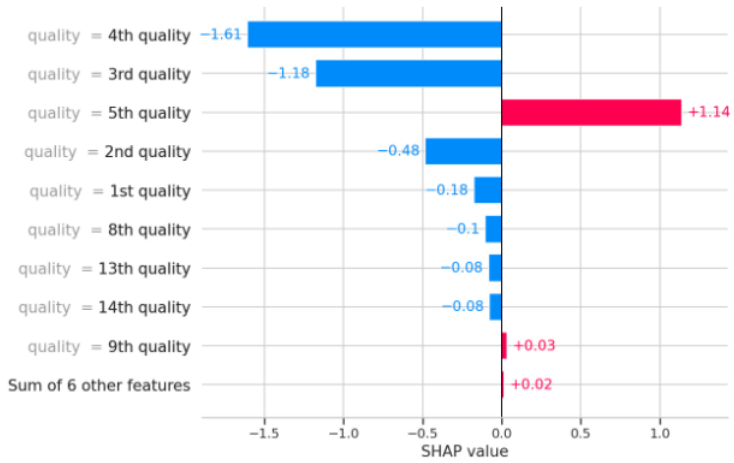

Figure 32: Importance of the word "quality"

8 Conclusion

In this paper we presented different models applied on sentiment analysis and classification, those models try to highlights the fast evolution that were made in a short amount of time in sentiment analysis with the help of deep learning, as we can see, state of the art models based on deep learning transformers completely outperform rule based and machine learning methodologies.

It is also interesting to note that recent improvement didn't only try to increase accuracy, but also to reduce training time and other weaknesses related to sequence to sequence models.

BERT is now a building block for all recent models on NLP task such as ROBERTA and ALBERT, the attention mechanism clearly plays a center role in all those models.

References

[1] R. Adarsh, A. Patil, S. Rayar, and K.M. Veena. Comparison of vader and lstm for sentiment analysis. International Journal of Recent Technology and Engineering, 7:540–543, 03 2019.

[2] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive recursive neural network for target-dependent twitter sentiment classification. 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference, 2:49–54, 06 2014.

[3] C.J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014, 01 2015.

[4] Mateusz Lango. Tackling the problem of class imbalance in multi-class sentiment classification: An experimental study. Foundations of Computing and Decision Sciences, 44:151–178, 06 2019.

[5] Harnani Mat Zin, Masrah Murad, and Nurfadhlina Sharef. The effects of pre-processing strategies in sentiment analysis of online movie reviews. AIP Conference Proceedings, 1891:020089, 10 2017.

[6] Manish Munikar, Sushil Shakya, and Aakash Shrestha. Fine-grained sentiment classification using bert. 2019.

[7] Sharnil Pandya and Pooja Mehta. A review on sentiment analysis methodologies, practices and applications. 10 2020.

[8] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. EMNLP, 14:1532–1543, 01 2014.

[9] Michael Phi. Illustrated Guide to LSTM's and GRU's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, 2018. [Online; accessed 24-sep-2018].

[10] Nikil T and Aloysius Amalanathan. Data preprocessing in sentiment analysis using twitter data. 3:89–92, 07 2019.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[12] Oinkina with Hakyll. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. [Online; accessed August 27, 2015].

[13] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. pages 1480–1489, 01 2016.

[14] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis : A survey. 2018.