Sentiment Analysis on Booking.com Reviews

Course: Machine Learning and Content Analytics

Professors: Mr. Papageorgiou & Mr. Perakis

Students:

Maria-Eleni Tsoutsa (p2822134)

Glykeria Eri Kaimenopoulou (p2822114)

Ioannis Nikas (p2822125)

Konstantina Vioni (p2822107)

Contents

Abstract

During the last 10 years there is a tremendous progress in the NLP field. Our goal is to compare the performance of different NLP models and approaches. We will work with a Kaggle dataset that refers to Booking.com reviews and the objective of the project is to perform a sentiment analysis. The analysis will be done on a balanced dataset and metric of interest is the accuracy. We will compare different models and we will keep the best one. The input of the model will be a Booking.com review and the output will be the probability of it being positive.

The graphs below show the evolution of the NLP models and the objective of this project is to test, compare and evaluate some of these algorithms on actual data.

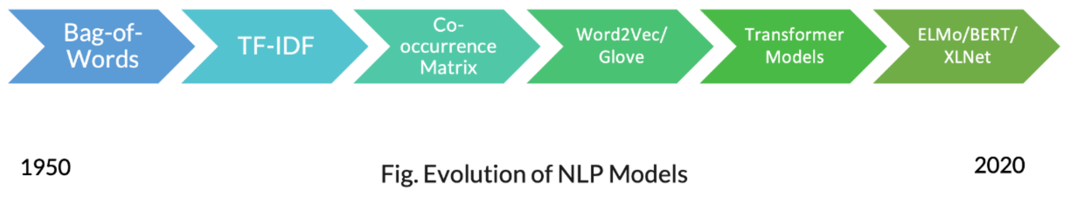


Figure 1

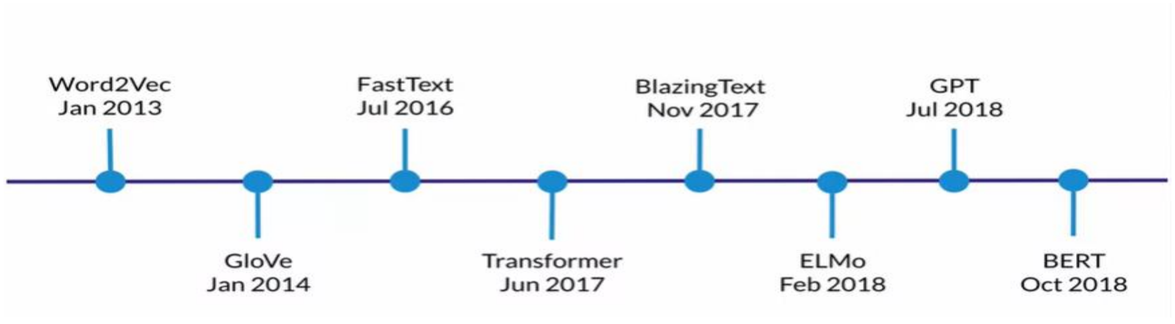


Figure 2

- January 2013: Word2Vec algorithm
- January 2014: GloVe algorithm
- July 2016: FastText algorithm
- June 2017: Transformer architecture, “Attention Is All You Need”
- November 2017: BlazingText algorithm
- February 2018: ELMo algorithm
- July 2018: GPT model architecture
- October 2018: BERT model architecture

Business Case

The main scope of the project is to help the owners of a hotel understand if the feedback from the visitors is positive. More specifically, a hotelier using the reviews/comments from Facebook, e-booking, or other sites and social media platforms could find the probability of the reviews of their visitors being positive. Having a high score means a great probability of the feedback being positive. On the other hand, a lower score would lead to neutral or negative feedback which means that some services or features should be evaluated and/or change.

This is a very useful tool to utilize comments and/or reviews being made in several platforms and comprehend how well business is going. The entrepreneurs, on their own, could find whether their clients are satisfied or not with two simple clicks by exporting and uploading their reviews. So, they can gain more insights and act if an evaluation of a service or an improvement is needed.

Project Goal

Our goal is to evaluate different NLP such as:

- Bag of Words with TF-IDF and Logistic Regression
- LSTM with word Embeddings
- A Pre-Trained DistilBERT model with Transformers
- A Fine-Tuned Pre-Trained DistilBERT model with Transformers

The Dataset

The dataset contains 515,000 customer reviews of luxury hotels across Europe and it contains 17 fields. However, for this project we will focus on the following three fields:

- **Negative_Review**: Negative Review the reviewer gave to the hotel. If the reviewer does not give the negative review, then it should be: 'No Negative'
- **Positive_Review**: Negative Review the reviewer gave to the hotel. If the reviewer does not give the negative review, then it should be: 'No Positive'
- **Reviewer_Score**: Score the reviewer has given to the hotel, based on his/her experience

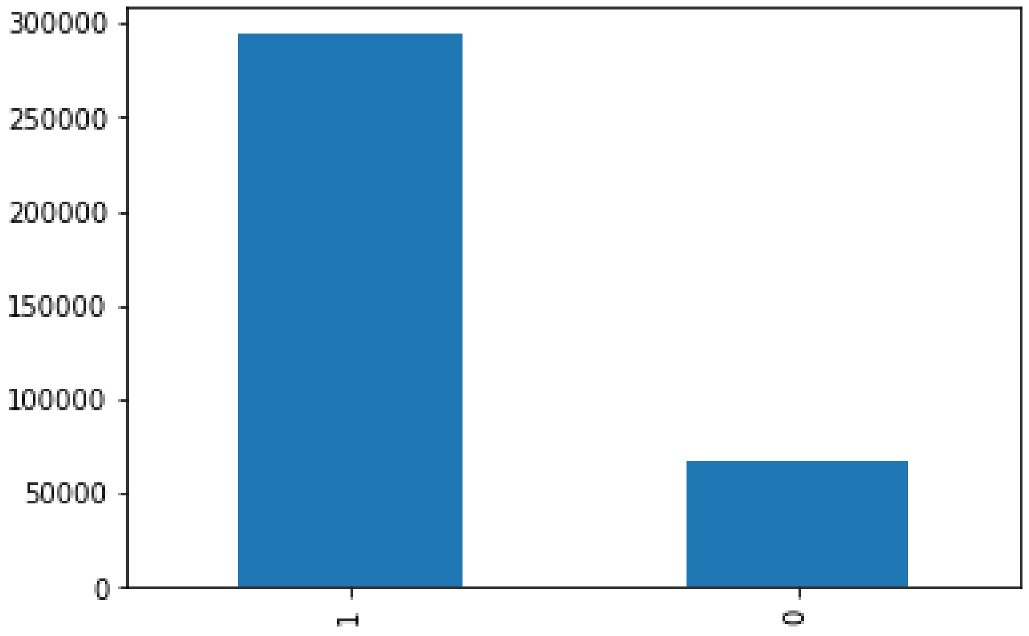
For this analysis we will concatenate the "negative" and "positive" reviews creating a new field called "review". Also, for the "reviewer_score" we will apply the transformation below:

- 1: Positive reviews are those with a score greater than or equal to **8.5**
- 0: Negative reviews are those with a score less than or equal to **6.5**

NB: We will ignore are the reviews between **6.5 and 8.5** considering them as neutral.

Data Preparation

As mentioned earlier, only the truly positive and negative reviews will be kept by removing those where lies between 6.5 and 8.5. After removing the "neutral" we ended up with 362K observations, but we were dealing with an unbalanced case where the 80% of the observations were labelled as "1" and the rest 20% as "0".



Under-Sampling

Clearly, we are dealing with an unbalanced case. There are many approaches to this case, such as reporting the F-1 Score or the ROC AUC Score but we want to keep it more straightforward and to report the accuracy of the different models. For that reason, we will apply a technique called "under-sampling" where in essence we will remove samples from the majority class without replacement. In this way we will alleviate imbalance in the dataset. On the other hand, since always there is a trade-off, by applying this technique we may increase the variance of the classifier and lose important information.

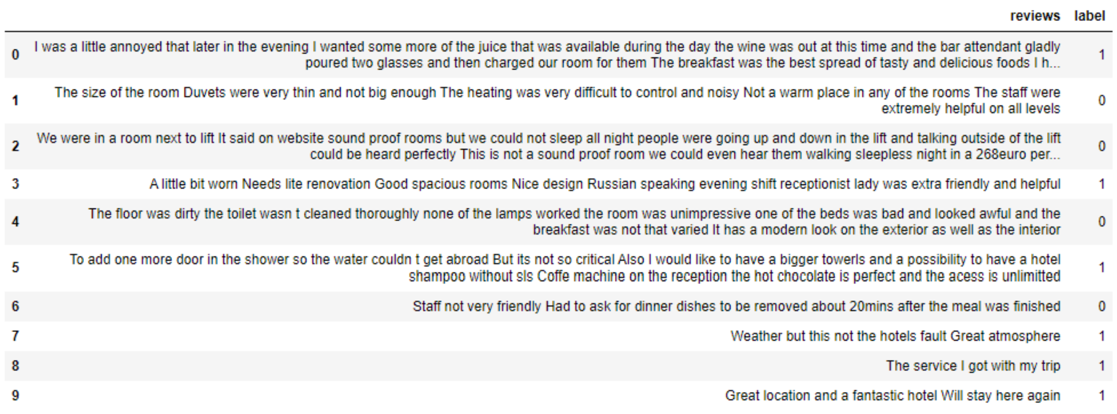
After applying under-sampling, we ended up with a balanced dataset of **135998** observations where the 67999 are positive and the 67999 are negative reviews.

Text Replacing

According to the column descriptions, if a positive review is blank, then the corresponding value of the cell is "No positive" and if a negative review is blank, then the corresponding value of the cell is "No negative". For that reason, we replaced these values with an empty string. Finally, we ignored the 66 empty reviews.

Reviews Concatenation

Finally, we concatenated the "Positive Review" and the "Negative Review" column by creating a new column called "**reviews**". Below we represent a sample of the final dataset.



	reviews	label
0	I was a little annoyed that later in the evening I wanted some more of the juice that was available during the day the wine was out at this time and the bar attendant gladly poured two glasses and then charged our room for them The breakfast was the best spread of tasty and delicious foods I h...	1
1	The size of the room Duvets were very thin and not big enough The heating was very difficult to control and noisy Not a warm place in any of the rooms The staff were extremely helpful on all levels	0
2	We were in a room next to lift It said on website sound proof rooms but we could not sleep all night people were going up and down in the lift and talking outside of the lift could be heard perfectly This is not a sound proof room we could even hear them walking sleepless night in a 268euro per...	0
3	A little bit worn Needs lite renovation Good spacious rooms Nice design Russian speaking evening shift receptionist lady was extra friendly and helpful	1
4	The floor was dirty the toilet wasn t cleaned thoroughly none of the lamps worked the room was unimpressive one of the beds was bad and looked awful and the breakfast was not that varied It has a modern look on the exterior as well as the interior	0
5	To add one more door in the shower so the water couldn t get abroad But its not so critical Also I would like to have a bigger towerls and a possibility to have a hotel shampoo without sls Coffe machine on the reception the hot chocolate is perfect and the acess is unlimitted	1
6	Staff not very friendly Had to ask for dinner dishes to be removed about 20mins after the meal was finished	0
7	Weather but this not the hotels fault Great atmosphere	1
8	The service I got with my trip	1
9	Great location and a fantastic hotel Will stay here again	1

Train and Test Dataset

For the Machine Learning models, we split the dataset into train (75%) and test (25%) datasets. We will train the models on the train dataset, and we will report the accuracy on the test dataset enabling us to compare the performance of the different models and approaches.

Exploratory Data Analysis

We can get some insights about the reviews by getting some descriptive statistics.

Number of Words per Review Type

Since we are dealing with text data, that are unstructured data, we should convert them to structured data. As a first step, we will represent the number of words by review type.

As we can see from the table below, the negative reviews tend to contain more words compared to positive reviews (48 vs 29 tokens on average). This makes sense, since when someone leaves a negative review tends to explain his/her complaints. On contrary, when it comes to a positive review, customers tend to be more concise.

```
           count        mean        std  min   25%   50%   75%    max
label
0        67999.0   48.285813  54.286102  1.0  14.0  31.0  61.0  689.0
1        67999.0   29.733437  32.479074  1.0  10.0  20.0  38.0  605.0
```

Most Common Words

At this point we will get the most common words of the positive and negative reviews respectively. For this report we will remove the most common English words such “the”, “a”, “to”, “of” and so on. This group of words are known as stopwords and it is a common technique to be removed when we want to build an NLP Machine Learning model.

Let’s see the 25 most common words in positive reviews:

```
('staff', 34222)
('room', 32656)
('location', 25941)
('hotel', 25808)
('great', 19198)
('breakfast', 18137)
('good', 15518)
('friendly', 14317)
('helpful', 13075)
('excellent', 11473)
('clean', 10517)
('nice', 10433)
('comfortable', 9928)
('bed', 8987)
('rooms', 8422)
('nothing', 8138)
('stay', 7588)
('everything', 6748)
('would', 6598)
('lovely', 6513)
('really', 6406)
('service', 5946)
('perfect', 5459)
('small', 4946)
('could', 4929)
('bar', 4926)
```

Let’s see the 25 most common words in negative reviews:

```
('room', 66918)
('hotel', 35105)
('staff', 26256)
('location', 25922)
('breakfast', 17996)
('good', 16354)
('small', 14578)
('bed', 14058)
('rooms', 13963)
('one', 9802)
('bathroom', 9191)
('night', 8740)
('reception', 8159)
('us', 8136)
('service', 7981)
('like', 7945)
('would', 7784)
('poor', 7422)
('clean', 7333)
('nice', 7284)
('could', 7000)
('shower', 6964)
('stay', 6897)
('even', 6809)
('get', 6568)
('also', 6022)
```

As we can see, in the positive reviews we see words such as “friendly”, “nice”, “clean”, “excellent”, “helpful” and so on, where in negative reviews we see words such as “small”, “reception”, “even” etc. However, there are some words that appear in both positive and negative reviews, that is why it makes sense to get the ratio of these words.

Sentiment Score of the Words

In NLP tasks and in sentiment analysis we define the word’s sentiment score as the of the times that appear in positive and negative reviews. So, the words with the highest positive sentiment score are those that tend to appear in the positive reviews but not in the negative and vice versa for the words with the highest negative sentiment score. Let’s have a look at the words with the highest positive and negative sentiment score with their corresponding score:

Top 25 Words with a Positive Sentiment Score

```
[('gem', 20.166666666666668),
 ('spotlessly', 16.583333333333332),
 ('immaculate', 14.5),
 ('loved', 14.212121212121213),
 ('amazingly', 12.583333333333334),
 ('dislike', 11.363636363636363),
 ('wonderful', 11.294444444444444),
 ('fabulous', 11.218390804597702),
 ('gorgeous', 10.25),
 ('impeccable', 10.181818181818182),
 ('superb', 10.163934426229508),
 ('memorable', 9.846153846153847),
 ('favourite', 9.692307692307692),
 ('spotless', 9.375),
 ('notch', 9.181818181818182),
 ('beautifully', 9.023809523809524),
 ('fab', 8.823529411764707),
 ('exceptional', 8.81132075471698),
 ('amazing', 8.801960784313726),
 ('outstanding', 8.763636363636364),
 ('delicious', 8.553398058252426),
 ('fantastic', 7.961538461538462),
 ('stunning', 7.744186046511628),
 ('picky', 7.411764705882353),
 ('brilliant', 6.654545454545454)]
```

Top 25 Words with a Negative Sentiment Score

```
[('impolite', 130.0),
 ('bugs', 89.0),
 ('unclean', 65.0),
 ('incompetent', 53.5),
 ('filthy', 50.0),
 ('worst', 43.833333333333336),
 ('claimed', 41.333333333333336),
 ('unfriendly', 39.904761904761905),
 ('peeling', 39.75),
 ('unhelpful', 36.416666666666664),
 ('stank', 34.333333333333336),
 ('dirty', 33.46153846153846),
 ('supposedly', 30.0),
 ('falling', 29.666666666666668),
 ('disgusting', 29.157894736842106),
 ('pathetic', 28.25),
 ('arrogant', 27.25),
 ('unprofessional', 25.53846153846154),
 ('appalling', 25.363636363636363),
 ('disgrace', 24.2),
 ('shocking', 23.875),
 ('tatty', 23.6),
 ('compensation', 23.153846153846153),
 ('blood', 22.166666666666668),
 ('smelly', 21.956521739130434)]
```

With this approach we get a clear picture. Words like “fabulous”, “gorgeous”, “superb”, “amazing”, “fantastic” and so on have a very high positive sentiment score and words like “bugs”, “impolite”, “unclean”, “worst”, “dirty” and so on get a very high negative sentiment score.

Word Clouds on the Sentiment Score

The above results can be represented with a Word Cloud. Below we show the Word Cloud of the words with the higher positive sentiment score and that one of the higher negative sentiment score.

Higher Positive Sentiment Score

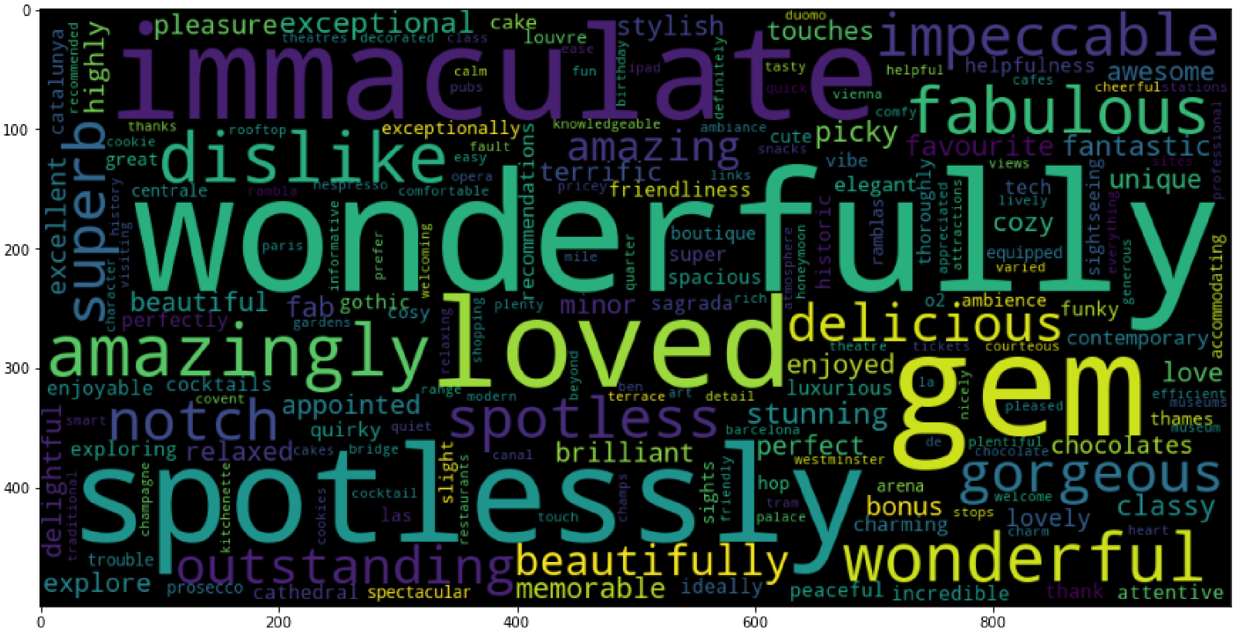


Figure 3

Higher Negative Sentiment Score

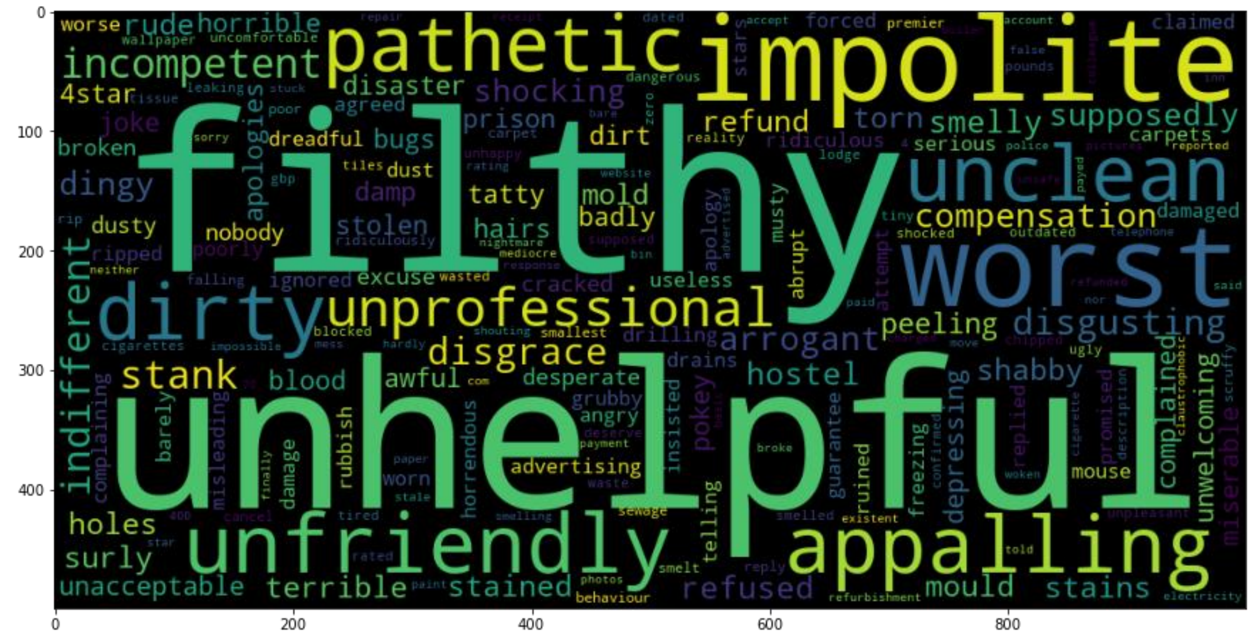


Figure 4

Logistic Regression Model

Logistic regression is an example of supervised learning and classification algorithm. It is used to assign observations to a discrete set of classes and transforms its output using the logistic sigmoid function to return a probability value. In our project, we will use logistic regression to classify the reviews that are positive or not.

We have already created the train and the test dataset. At this point we will build the logistic regression model using the Scikit-Learn Python library. For the model we will follow the next steps below.

Data Cleaning and Text Mining

The reviews are plain texts which are unstructured data. Before we convert them to structured data, we will need to do some data cleansing such as:

- Convert the text to a lower case
- Remove punctuation
- Remove the stop words
- Regular Expression for the tokenizer: It selects tokens of 2 or more alphanumeric characters (punctuation is completely ignored and always treated as a token separator)

Term-Document Matrix

The next step is to create the term document matrix. Taking as input the cleaned reviews, we will get the vocabulary, known as vector space. Then, for each document (i.e. for each review) we will get the TF-IDF. We want to ignore the relatively sparse tokens, and for that reason we will consider the ones that have appeared at least 5 times in the data. The Term-Document Matrix will be the features of the classification model.

Results

Keeping in mind that we are dealing with a balanced dataset, we evaluate the models by taking into account the accuracy as a KPI. In the train dataset the accuracy of the model was **90%** and in the test dataset was **88%.**

Classification Report of the Train Dataset

```
             precision    recall  f1-score   support

           0       0.90      0.90      0.90     50892
           1       0.90      0.90      0.90     51106

    accuracy                           0.90    101998
   macro avg       0.90      0.90      0.90    101998
weighted avg       0.90      0.90      0.90    101998
```

Classification Report of the Test Dataset

```
           0       0.88      0.87      0.88     17107
           1       0.87      0.88      0.87     16893

    accuracy                           0.88     34000
   macro avg       0.88      0.88      0.88     34000
weighted avg       0.88      0.88      0.88     34000
```

LSTM Model

Long short-term memory is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network (RNN) can process not only single data points (such as images), but also entire sequences of data (such as speech or video). The purpose of using it is that the dataset at hand is raw text reviews.

Model Architecture

The next model that we will try is the LSTM with Embeddings. We have tried different models and the final one has the following architecture.

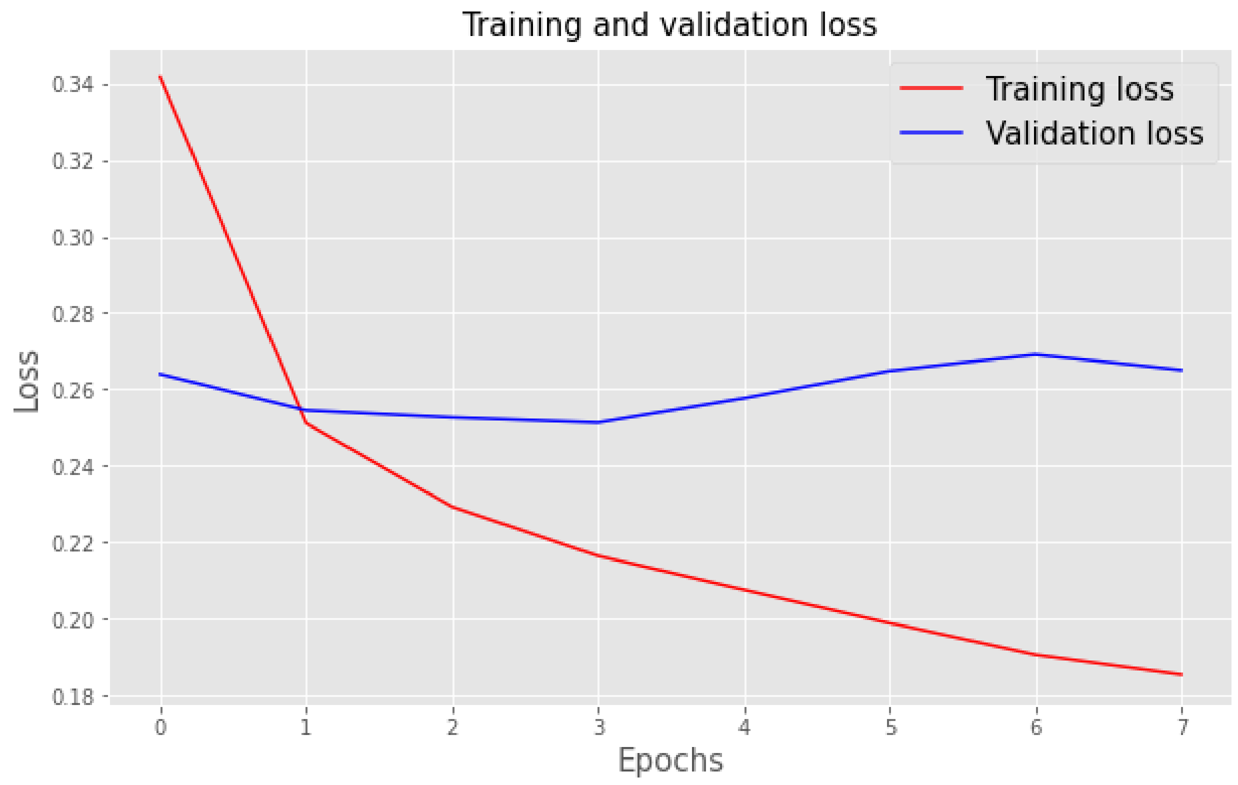
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_2 (Embedding)     (None, None, 64)          640000    
                                                                 
 bidirectional_4 (Bidirectio  (None, None, 128)        66048     
 nal)                                                            
                                                                 
 bidirectional_5 (Bidirectio  (None, 32)               18560     
 nal)                                                            
                                                                 
 dense_4 (Dense)             (None, 64)                2112      
                                                                 
 dropout_2 (Dropout)         (None, 64)                0         
                                                                 
 dense_5 (Dense)             (None, 1)                 65        
                                                                 
=================================================================
Total params: 726,785
Trainable params: 726,785
Non-trainable params: 0
```

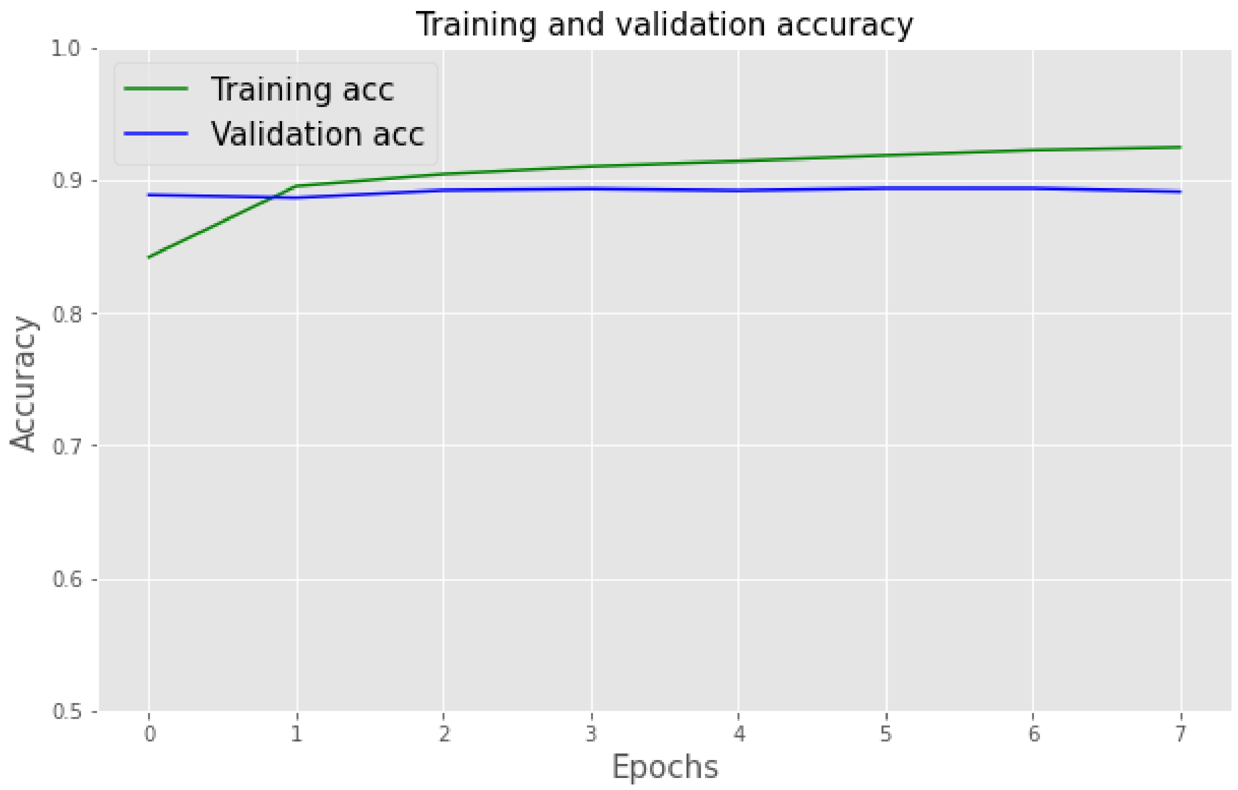
As we can see, we started with an embedding layer of 64 dimensions. Then we added a bi-directional LSTM layer of 128 layers, then another bi-directional LSTM layer of 32 layers, then a dense layer of 64 with a ReLU activation function, then added a dropout layer with 50% and finally, we added a dense layer of 1 layer and a sigmoid activation function.

We trained the model using the Adam optimizer with 0.0001 learning rate. The batch size was set to 30 and the epochs were set to 10. However, due to the “early stopping” of patience 2, the optimum model required only 7 epochs. Below we represent the loss and the accuracy of the train and test dataset for each epoch.

```
Epoch 1/10
3400/3400 [==============================] - 131s 37ms/step - loss: 0.3417
- accuracy: 0.8418 - val_loss: 0.2639 - val_accuracy: 0.8887
Epoch 2/10
3400/3400 [==============================] - 126s 37ms/step - loss: 0.2511
- accuracy: 0.8952 - val_loss: 0.2544 - val_accuracy: 0.8866
Epoch 3/10
3400/3400 [==============================] - 126s 37ms/step - loss: 0.2292
- accuracy: 0.9043 - val_loss: 0.2526 - val_accuracy: 0.8922
Epoch 4/10
3400/3400 [==============================] - 126s 37ms/step - loss: 0.2164
- accuracy: 0.9102 - val_loss: 0.2512 - val_accuracy: 0.8933
Epoch 5/10
3400/3400 [==============================] - 126s 37ms/step - loss: 0.2075
- accuracy: 0.9141 - val_loss: 0.2576 - val_accuracy: 0.8920
Epoch 6/10
3400/3400 [==============================] - 126s 37ms/step - loss: 0.1988
- accuracy: 0.9183 - val_loss: 0.2647 - val_accuracy: 0.8937
Epoch 7/10
3400/3400 [==============================] - 126s 37ms/step - loss: 0.1905
- accuracy: 0.9223 - val_loss: 0.2691 - val_accuracy: 0.8936
Epoch 8/10
3400/3400 [==============================] - 125s 37ms/step - loss: 0.1853
- accuracy: 0.9245 - val_loss: 0.2650 - val_accuracy: 0.8912
```

We can better grasp the information above graphically as follows.


Training and validation loss
Training loss
Validation loss
Loss
Epochs
0.34
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0
1
2
3
4
5
6
7


Training and validation accuracy
Training acc
Validation acc
Accuracy
Epochs
1.0
0.9
0.8
0.7
0.6
0.5
0
1
2
3
4
5
6
7

Results

The accuracy of the model on the train dataset was **92.68%** and on the test dataset **89.37%**. Below we represent the classification report for the train and test dataset.

Train Dataset

```
              precision    recall  f1-score   support

           0       0.90      0.96      0.93     50892
           1       0.96      0.89      0.92     51106

    accuracy                           0.93    101998
   macro avg       0.93      0.93      0.93    101998
weighted avg       0.93      0.93      0.93    101998
```

Test Dataset

```
              precision    recall  f1-score   support

           0       0.87      0.93      0.90     17107
           1       0.93      0.85      0.89     16893

    accuracy                           0.89     34000
   macro avg       0.90      0.89      0.89     34000
weighted avg       0.90      0.89      0.89     34000
```

Thus, based on the results, the LSTM model is better than the logistic regression.

Pre-Trained Model with Transformers

At this point we will test the performance of a pre-trained model that was built using transformers and more particularly, we will work with the DistilBERT base model (uncased), fine-tuned on SST-2. BERT is at its core a transformer language model with a variable number of encoder layers and self-attention heads.

For a valid comparison, we will report the accuracy of the DistilBERT model on the same test dataset that we have used for the other two models. Bear in mind that since we will work with a pre-trained model, we will not train any model, and as a result, we will not report the accuracy on the train dataset.

Issues with the Data and the Model

The original reviews received from the Kaggle are somehow cleaned since the punctuation has been removed. This incident affects the model since it expects sentences as input. On contrary, it treats the whole review as a single sentence. Moreover, the maximum length of tokens for each sentence is 512, and in our data, there were 4 reviews that exceed this limit. In order to overcome this issue, we labeled them as "negative" reviews. Alternatively, we could have truncated the input review to 512 tokens.

Results

The accuracy of the pre-trained model on the test dataset was **79%** which is much worse than the previous models.

```
              precision    recall  f1-score   support

           0       0.77      0.84      0.80     17107
           1       0.82      0.74      0.78     16893

    accuracy                           0.79     34000
   macro avg       0.79      0.79      0.79     34000
weighted avg       0.79      0.79      0.79     34000
```

The next step is to fine-tune the existing pre-train model based on the train dataset.

Fine-Tuned Transformers

The final step was to fine-tune the DistilBERT model. We run just a single epoch and we saved the model of the last checkpoint. The reason why we only run a single epoch is due to overfitting.

Results

The fine-tuned model with transformers was the clear winner since the accuracy of the model on the test dataset was **91.8%**!

Accuracy of the model on the test dataset

```
              precision    recall  f1-score   support

           0       0.91      0.93      0.92     17107
           1       0.92      0.91      0.92     16893

    accuracy                           0.92     34000
   macro avg       0.92      0.92      0.92     34000
weighted avg       0.92      0.92      0.92     34000
```

Conclusion & Future work

According to the results, the best model was the fine-tuned pre-trained model. The results were not a surprise since we expected that the advanced transformers will outperform the rest models.

Because of time restrictions and technical problems, we could not test more models for this project. This is an addition that we should do in the future to create a more accurate model. Also, the upcoming project would be to create a model which could connect a positive review with specific services or features. This would complete the project because we could provide a full consulting service for hoteliers.

Appendix

The models were run on the Google Colab using Python. For the LSTM and the Transformers models, we worked on a GPU runtime. For the LSTM we worked with TensorFlow and for the Transformers we worked with the transformers library by HuggingFace. We have provided all the notebooks that are reproducible. Finally, we have shared the fined-tuned model and how to load it.

Time plan & Team roles

We started the assignment in the beginning of July. The first couple of weeks of July we were contemplating on business case and the appropriate data for the case. Our initial thought was to build a recommendation engine for hotels based on their features. However due to implementation difficulties that came along the way, we changed our case to a sentiment analysis based on hotel reviews. By the third week of July we had our final business case, the dataset and had started building the algorithm. During August we developed the three models that have been included in this present report. Finally in the first week of September we put together the report along with the GitHub deliverables.

As far as the team roles, all members of the team brought knowledge and ideas on the project. However, a small differentiation regarding the effort existed in two parts of the assignment:

- The report was mainly executed by Maria Eleni Tsoutsa and Glykeria Eri Kaimenopoulou
- The data preparation was mainly executed by Konstantina Vioni and Ioannis Nikas

The core code was developed by all members of the team equally.