# Língua Natural - MP2 (Hotel Review Classifier)

Grupo 15
Hugo Ribeiro (95590)
Murillo Teixeira (105038)

October 27, 2023

## 1 Introduction

Our project revolves around a dataset containing 1400 hotel reviews, evenly distributed (as shown in Fig. 1) among four categories: Deceptive Negative (DN), Deceptive Positive (DP), Truthful Positive (TP), and Truthful Negative (TN) . These categories differ in vocabulary, review length, and the type of information presented. The key focus is on distinguishing between these 4 categories.

To tackle this, we employed four different models, each increasing in complexity, with a special focus on the RoBERTa model, which, initially, we believed would yield better results. These models were tested using a train-test split, and the results were compared to assess which one performed best.

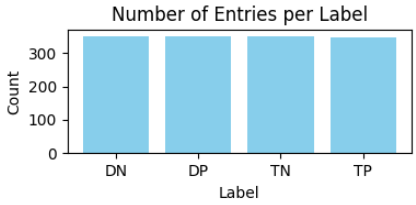


Figure 1: Entries distribution among classes

## 2 Models

### 2.1 Multinomial Logistic Regression

The initial approach chosen was Multinomial Logistic Regression, which is very simple and acted as a baseline for the following models. The first step was to create a matrix of token counts per review, which was done using scikit-learn's CountVectorizer class, to transform the text into vectors. After this, the LogisticRegression class from the same library was used to carry out the classification.

Multinomial logistic regression is an extension of logistic regression used for multiclass classification. It models the probability of an instance belonging to each class from the data generated by the vectorization process and selects the class with the highest probability as the prediction.

### 2.2 Deep Neural Network

The second method implemented was a simple Deep Neural Network (DNN). To implement it, the first step was to preprocess the reviews and labels. Reviews needed to be tokenized with the Keras *Tokenizer* class. Initially, it assigned a unique number to each token (word), and then, for each review, created a vector where each token was replaced by its corresponding number. After that, the sequences went through a padding process to standardize the vector sizes. For the labels the One Hot Encoder was used, transforming each output into a vector of size 4, which could then be compared with the 4 probabilities calculated as the output of the neural network.

The model consists of 4 sequential layers: the first one performs input embedding, capturing the semantic relationship between words by converting the token vector into another vector of fixed dimensions. The next layer, GlobalMaxPool1D, reduces the dimensionality of the information by selecting only the most relevant features. The subsequent layers are Dense; the first introduces non-linearity with a ReLU activation function, and the last one uses softmax to generate probabilities for belonging to each class. The model was compiled with the Adam optimizer and categorical cross-entropy loss function, ideal for multi-class classification.

### 2.3 RoBERTa

The main model employed for this project is based on the RoBERTa architecture, a state-of-the-art transformer-based model for NLP tasks. RoBERTa builds on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates.

The RoBERTa model used in this project is loaded from the "roberta-base" pre-trained model. This pre-trained model has been fine-tuned for a variety of NLP tasks, so we thought it would make it suitable for this task. To implement it we followed a notebook found on the Hugging Face website and changed it to suit our needs.

To make predictions, the model has a linear layer followed by a ReLu activation function. This maps the hidden state representation generated by RoBERTa to the number of labels we have (4). This

transformation is what allows the model to make predictions.

The only pre-processing needed was tokenization, similar to the one from the DNN, but using using the RoBERTa tokenizer. Both the loss function and the optimizers used were the same from the DNN method.

### 2.4 GPT-3.5-Turbo

GPT-3.5 models can "understand" and generate natural language or code, so we thought it would be an interesting idea to use them for this classification task.

The GPT-3.5-Turbo is the most capable GPT-3.5 model and is the most recent one that can be fine-tuned, hence our decision to use this model, because we wanted to compare the results before and after fine-tuning. This approach allowed us to assess the impact of fine-tuning on the model's effectiveness for our classification task.

For the fine-tune process, we used the OpenAi API which simplifies the task. We just needed to convert the test and train data to the required format (prompts). Once the pre-processing was complete, we initiated the fine-tuning, which involved training over the course of 3 epochs. It's worth noting that fine-tuning on the OpenAI platform resulted in a cost of 8€. This is due to the fact that GPT-3 models provided by OpenAI are not available for free.

## 3 Experimental Setup and Results

Every model was tested using a Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71GHz processor.

As the first two models (Multinomial Logistic Regression and DNN) required fewer resources and were quickly trained, cross-validation with the labeled data was used to evaluate them with sklearn's KFold class and $k = 10$. For the last two (RoBERTa and GPT-3.5) it becomes very expensive to perform cross-validation with the hardware we have, so we only performed a 90%-10% split train-test with the labeled data.

All of the model's performances were evaluated using the accuracy of the predictions, that can be seen in Tab. 1. And, for DNN and RoBERTa, we can see their confusion matrices in Fig. 2 for one train-test split for further analysis.

| | |
|---|---|
| Logistic Regression | 0.786 |
| Deep Neural Network | 0.754 |
| RoBERTa | 0.964 |
| GPT-3.5-Turbo 2 | 0.300 |
| GPT-3.5-Turbo (Fine-tuned) | 0.593 |

Table 1: Accuracy for each model

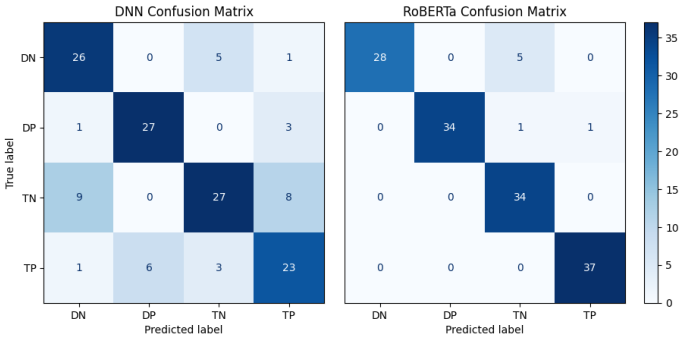


Figure 2: DNN and RoBERTa's Confusion Matrices

## 4 Discussion

Our RoBERTa based model did not have such a hard time with this task has it only missclassifies very few instances, but a more detailed test, with cross-validation, should be made to confirm result.

The greatest challenge, for every model, was to differentiate between Truthful and Deceptive. This can be easily seen in the confusion matrices: for each row, the most common incorrect guess is just a switch between Truthful and Deceptive. Even for RoBERTa, here is a example of a sentence where RoBERTa made this mistake:

"TRUTHFULPOSITIVE Just spent three nights at the Amalfi Hotel, while in Chicago on business. Booked a king-corner room and loved it [...] If you want a great hotel (at a great price) right in the thick of everything, stay at the Amalfi."

For GPT-3.5-Turbo, with and without fine-tuning, the same situation happened. When we tested the original model only 15 out of 140 were classified as Deceptive, when this value should be closer to 70. The prompting may be one of the causes, both of which can have a significant impact on the model's performance.

## 5 Future work

About the challenge in differentiating Truthful and Deceptive, the first thing that could be tested is to use two different classifiers: one to differentiate between Truthful and Deceptive and another to differentiate between Positive and Negative (which appears to be easier).

Apart from this the results were positive but could be better for GPT-3.5 and DNN, possibly due to limited data. Future improvements could involve acquiring more data, exploring diverse DNN architectures like CNNs and fine-tuning GPT-3.5 with additional epochs.

Other important improvements but that would need better hardware would be optimizing hyperparameters, and implementing cross-validation for a comprehensive evaluation of every model's performance.