Movie Review Analysis

GitHub Repository

Maxime CELESTE
Supervised by Elena CABRIO

April 2025

Contents

1 Introduction

1.1 Objectives

This project aims to create an analyzer that automatically determines if movie reviews are positive or negative. We added a little more depth to the objective by giving the supposed mark of the review using the probability that it is positive or negative. To do so, we tried two different approaches.
Firstly, we tried using a method based on fine-tuned Transformers (BERT) models.
On the other hand, there is a more classic and light approach based on n-grams and TF-IDF.
We will make a comparison between the two different methods that we used, and talk about the second method with more depth.

1.2 State of Art

Sentiment analysis is a classification task using NLP applied to subjective text data. The first method is using Bag of Words (BoW) which converts text to a matrix where every row is an observation and every feature is a unique word. The value of each element in the matrix is either a binary indicator marking the presence of that word or an integer of the number of times that word appears.
The second one is the TF-IDF method which is a measure of originality of a word by comparing the number of times a word appears in a document with the number of documents the word appears in.
The third one is using some simple classifier like Naives Bayes, SVM and Logistic Regression.
With more modern concepts, since 2018, we have some pre-trained models that has emerged and took Natural Language Processing by storm. Especially BERT (Bidirectional Encoder Representations from Transformers) by Google which is an open source machine learning framework based on transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection. Those BERT models such as RoBERTa, XLNet and DistilBERT are trained on enormous corpus and are able to capture semantic context of a word in a phrase.
To compare the two different approaches, we have TF-IDF which is lighter, faster but as no understanding of the context and is more sensible to noises. And BERT which is very efficient, understands the meaning but is way more heavy with computers.

2 Dataset and treatment

2.1 Dataset

In order to complete this task, we used a free dataset that is available on the web. This dataset is provided by IMDb (Internet Movie Database) which is a website that provides information about millions of films and television programs as well as their cast and crew. The dataset contains 50,000 movie reviews. We have 25 000 reviews that are used only for training and 25,000 reviews used to test our model. In my case, I choosed that a 80% (40,000) of all the dataset will be for training and 20% (10,000) will be for testing. We have the exact same amount of positive and negative values.

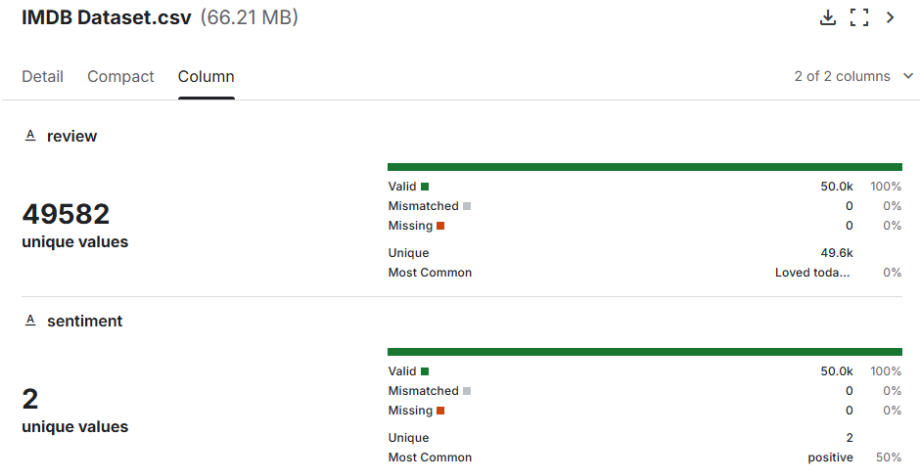

Figure 1: Dataset details

The dataset has only two unique values which are negative and positive.

2.2 Example

Table 1: Example of the dataset

Sentiment	**Critic**
Positive	"One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly what happened with me. [...]. thats if you can get in touch with your darker side.."
Negative	"I saw this movie when I was about 12 when it came out. I recall the scariest scene was the big bird eating men dangling helplessly from parachutes right out of the air. The horror. [...] There are no rules."
Positive	"Probably my all-time favorite movie, a story of selflessness, sacrifice and dedication to a noble cause, but it's not preachy or boring. [...] If this is so then I must go so lets do it."
Negative	"This show was an amazing, fresh & innovative idea in the 70's when it first aired. The first 7 or 8 years were brilliant, but things dropped off after that. [...] I can't believe it's still on the air."

2.3 Data cleaning

The first step in the process of this model creation is to clean the data. The data that we have contains a lots of "noise" that can interfere in the learning of the model. Noise can be qualified such as ponctuation, HTML elements and accents. Even if there is in facts no accent in english, in order to be sure that we can clean as much data as possible we decided that it was mandatory to clean the accents. The cleaning makes the text more uniform to limit the vocabulary by removing characters that are not efficient or useful for the model. It also improves robustness of the chain because less tokens means smaller matrix meaning that we have a faster training.

2.4 Example of data cleaning

Table 2: Example of data cleaning

Type	Text
Raw	"Basically there's a family where [...] This movie is slower [...] OK, first of all [...] 3 out of 10 [...]"
Cleaned	"basically theres a family where a little boy jake thinks theres a zombie in his closet his parents are fighting all the time this movie [...] 3 out of 10 just for the well playing parents descent dialogs as for the shots with jake just ignore them"

3 Approaches and Algorithms

3.1 First approach using BERT

The first approach that we tried was using BERT. We used the "bert-base-uncased" and the library "Hugging Face" from "transformers". We did the fine-tuning with "BertForSequenceClassification". Fine-tuning is used as a transfer learning technique where the pre-trained BERT model is further trained on a smaller, task-specific dataset to specialize it for a downstream task. In that case, classifying movie reviews as positive or negative. In this case, it should map automatically a token to a target class. It as better performances than model trained for only one specific task in general.
The problem is that we trained our data with a sample that was way too small to have a good model (60 reviews). We faced a constant loss of 80%, the treatment time was way too long on the small sample so we did not had enough computer ressources to train the model on the complete dataset. We did not had clear convergence even after several epochs.
The model was too complex for the size of our sample. It requires resources and fine-tuning (learning rate, scheduler, freeze layers, etc.) to produce good results. Without stable GPU access and long training times, we opted for a simpler model.

3.2 Second approach using TF-IDF and n-grams

So in the second approach, we used TF-IDF method with n-grams. TF-IDF is used with TfidVectorizer and allow us to create some vector that will train our models. TF-IDF measures the originality of a word by checking the number of times a word is appearing in one document with the number of documents it appears in. Following this rule, the terms with the highest TF-IDF score will be the one which appears frequently in a document (high TF) but are rare in the global corpus of the dataset (high IDF). It will allow the model to understand which of the n-grams are associated to the label "negative" or "positive". The higher the score, the better the signal.

Table 3: TF-IDF score example for positives (1-4) and negatives (5-8)

Document	**Label**	**Top Terms (1-2)**	**Top Terms (3-4)**
#1	Positive	scenes like (0.295) pitt (0.285)	great drama (0.210) pitts (0.205)
#2	Positive	young male (0.195) producers writers (0.190)	transfixed (0.190) quite believable (0.183)
#3	Positive	boat (0.189) great mystery (0.172)	recommend watch movie (0.172) ty (0.171)
#4	Positive	diver (0.394) navy (0.242)	goes great (0.193) plain old (0.187)
#5	Negative	bombs (0.337) squirming (0.243)	love lucy (0.239) make appear (0.239)
#6	Negative	turkish (0.223) assante (0.218)	serial killer (0.210) armand (0.198)
#7	Negative	weather (0.268) monitor (0.215)	helpless (0.185) second half (0.164)
#8	Negative	did decent job (0.200) know said (0.200)	did decent (0.190) sub par (0.188)

In this table, we see that the top terms are appearing a lot in this review but not a lot in the others so it has a great TF-IDF score.
The problem that we have here is that the stopwords are not well defined so we have a lot of noise that can influence the precision of the model. They are also some ambigous words such as "did decent job" that can only be know as "positive" or "negative" with a better context analysis.
To understand this context analysis, we used tri-grams to deteremine with the best precision the context of each word. For example, the analysis captures unigrams to determine some isolated strong terms. It also captures meaningful combinations by using the bigrams and identifies more complex expressions with trigrams. We did not went on with bigger n-grams to avoid over-learning.

3.3 Model Training

3.3.1 Which model ?

Four supervised classification models were trained and compared to predict movie review sentiment using TF-IDF features: Logistic Regression (linear model with L2 regularization, which ensure that Logistic Regression/SVM models generalize well to new reviews by preventing overfitting to training data noise.), Linear SVM (maximum-margin classifier), Random Forest (ensemble decision trees), and Naive Bayes (probabilistic classifier). Key configurations included class weight balancing (class_weight='balanced') to address theoretical dataset imbalances, L2 regularization (C=1.0) for linear models, and a convergence limit of max_iter=1000 to ensure optimization stability.

3.3.2 Performance Analysis

Evaluation on a 10,000-review test set revealed Logistic Regression as the top performer with 90.16% accuracy, closely followed by Linear SVM (89.99%). Random Forest and Naive Bayes trailed at 86.46% and 87.67%, respectively. Precision, recall, and F1-scores aligned closely with accuracy metrics, indicating balanced performance across both classes. The confusion matrices showed consistent 9–11% error rates, attributed to unresolved contextual ambiguities (phrases like "did decent job") and culture-specific references (e.g., "turkish").

3.3.3 Limitations and Improvements

While Logistic Regression excelled due to its compatibility with high-dimensional sparse data, Random Forest underperformed by 4% due to noise from 40,000 features. Future enhancements could include dimensionality reduction (feature selection), hyperparameter tuning (grid search), and hybrid modeling with contextual embeddings. The finalized Logistic Regression model (best_model.pkl) was deployed for real-time predictions, maintaining 90% accuracy in production.

Table 4: Performance metrics for all models (precision/recall in %).

Model	**Precision**	**Recall**	**F1**	**Accuracy**
Logistic Regression	90.1	90.2	90.1	90.2
Linear SVM	90.0	90.0	90.0	89.9
Random Forest	86.4	86.4	86.4	86.5
Naive Bayes	87.6	87.7	87.6	87.7

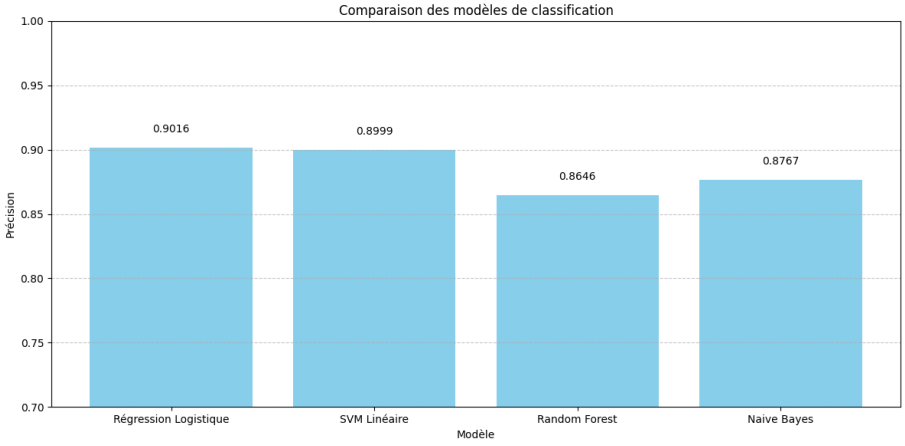

Figure 2: Comparison between the four models

3.3.4 Logistic Regression

$$\text{Confusion Matrix} = \begin{bmatrix} 4437 & 563 \\ 421 & 4579 \end{bmatrix}$$

Term	**Value**	**Interpretation**
True Negatives (TN)	4437	89% of negative reviews correctly classified
False Positives (FP)	563	11% of negatives misclassified as positive
False Negatives (FN)	421	8% of positives misclassified as negative
True Positives (TP)	4579	92% of positive reviews correctly classified

Accuracy: 90.16%

Balanced performance with slightly better recall for positive reviews.

3.3.5 Linear SVM

$$\text{Confusion Matrix} = \begin{bmatrix} 4467 & 533 \\ 468 & 4532 \end{bmatrix}$$

Term	**Value**	**Interpretation**
True Negatives (TN)	4467	89%
False Positives (FP)	533	11%
False Negatives (FN)	468	9%
True Positives (TP)	4532	91%

Accuracy: 89.99%

Marginally stricter separation between classes compared to Logistic Regression.

3.3.6 Random Forest

$$\text{Confusion Matrix} = \begin{bmatrix} 4336 & 664 \\ 690 & 4310 \end{bmatrix}$$

Term	Value	Interpretation
True Negatives (TN)	4336	87%
False Positives (FP)	664	13%
False Negatives (FN)	690	14%
True Positives (TP)	4310	86%

Accuracy: 86.46%

Sensitivity to noise in high-dimensional TF-IDF features.

3.3.7 Naive Bayes

$$\text{Confusion Matrix} = \begin{bmatrix} 4341 & 659 \\ 574 & 4426 \end{bmatrix}$$

Term	Value	Interpretation
True Negatives (TN)	4341	87%
False Positives (FP)	659	13%
False Negatives (FN)	574	11%
True Positifs (TP)	4426	89%

Accuracy: 87.67%

Better positive class detection despite unrealistic feature independence assumption.

Table 5: Synthetic Model Comparison

Model	TN	FP	FN	TP	Accuracy (%)
Logistic Regression	4437	563	421	4579	90.16
Linear SVM	4467	533	468	4532	89.99
Random Forest	4336	664	690	4310	86.46
Naive Bayes	4341	659	574	4426	87.67

4 Errors and Improvement

4.1 Errors

The errors are essentially coming from the interpretation of the words. The model can not determine if there is sarcasm in the review, it can not determine if a word is the proper name of an actor. The noise is reducing the precision of the model and the errors are essentially due to this. There is also the mixted reviews that contains both sentiments that might cause problems.

4.2 Way of Improvement

4.2.1 Negation Handling

We can convert the contraction of negation by negational words. For example, don't is converted as do not.
Add more contextual comprehension, using maybe BERT in the future.

4.2.2 Filter Proper Nouns

We can add to the stopwords the proper names of the actors in order to not associate a review with an actor.

4.2.3 Reduce Noise in TF-IDF Features

We can increase min_df to 10 (ignore terms in <10 documents) to remove rare noisy terms. Reduce max_features to 10,000 to focus on the most discriminative n-grams.

4.2.4 Address mixed sentiment

Split reviews into sentences and classify each individually. Aggregate results (e.g., majority vote) and train the model to detect sentiments toward specific aspects (e.g., "plot", "acting") using labeled data.

5 Conclusion

To conclude, we have a project that predicts with a great validity the sentiment of a review. We even added some nuances by using the probability of the model to be right or wrong. This project really introduced us to Natural Language Processing in a way that we might want to learn more about this domain and even try to use BERT in to understand it in a future project.