**Natural Language Processing Modern Techniques: An overview on the IMDB Large Movie Review and Amazon Reviews Datasets**

ENSAE Dakar

Mame Thierno NDIAYE

Abdou Hamid ALAGBE

ENSAE Dakar

Cheikh Sadibou NGOM

ENSAE Dakar

Introduction

In recent years, Natural Language Processing (NLP) has undergone remarkable advancements, driven by the development of deep learning architectures and the availability of large-scale datasets. These innovations have significantly improved the ability of machines to understand, interpret and generate human language. Recurrent Neural Networks, Long Short-Term Memory and Gated Recurrent Unit in particular, Transformer based architectures have been firmly established as state-of-the-art models in sequence modelling and almost all transduction problems. On many benchmarks, the transformer-based architecture has proven to be the best performing model (Lukasz Kaiser et al, 2017). Among the most prominent benchmarks for evaluating sentiment analysis are the IMDB large movie review (Andrew Y. Ng et al., 2011) and the Amazon reviews datasets which provide rich corpora of text both labeled and unlabeled with rating information.

This paper provides an overview of modern NLP techniques and their application to these two widely used datasets. We explore the evolution from traditional machine learning approaches, such as TF-IDF with logistic regression to the state-of-the-art deep learning models, including recurrent neural networks (RNNs), convolutional neural networks (CNNs), attention mechanisms, and more recently, transformer-based architectures like BERT and GPT. By comparing model performance and implementation challenges across the IMDB and Amazon datasets, we aim to highlight the importance of feature engineering in the process of constructing such architectures. In particular, we analyze how much preprocessing techniques applied on text data can impact the result of these models. Finally, we highlight the strengths and limitations of different techniques in real-world sentiment analysis tasks.

Ultimately, this overview serves as a foundation for understanding how contemporary NLP tools work in detail and how they can be applied to large-scale review data to derive meaningful insight and support a wide range of applications in recommendation systems, customer feedback analysis, and digital marketing strategies.

1. Natural Language Processing Pipeline

Understanding human language is not as simple for machines compared to humans. For a machine to understand a sequence (a boy of text, sentences), it has to model in a way it can comprehend. Humans understand texts, sequences but listening to each word and giving a meaning to these words internally. Natural Language Processing techniques use that method to model sequences.

**Tokenization**: Given sequence , the model parts it into small elements called tokens. This process is called tokenization. There are three principal tokenization techniques: character level tokenization, word-based tokenization and sub word-based tokenization. In character level tokenization, we break the sequence into individual characters composing it. This often reduces the size of the vocabulary[[1]](#footnote-1) but fails to grasp the meaning of understanding because the units are too small. And in fact, humans don’t break sequences to characters generally to understand a text. Thus, we will not test the character level tokenization in our work. In word-based tokenization, we divide the sequence into individual words composing it. These often leads to larger vocabulary size since related words are counted as different ones. For example, a word and its plural will be counted as two different words. This method grasps the meaning of words more efficiently than the previous. Lastly, the sub word tokenization divides sequences into words and words into its smaller parts such as prefix, root and suffix. It leads to smaller vocabulary size and solves the problems of word-based tokenization. In this work, we use the Byte Pair Encoding (BPE) as our sub word-based tokenizer.

**Vectorization**: This process often also called **Embedding** in modern techniques, gives meaning to the tokens in our vocabulary. Its role is to transform tokens into vectors that grasp the meaning of words on a fundamental level. There are multiple vectorization techniques: Term Frequency-Inverse Document Frequency (TF-IDF), Bag of words, Word2Vec etc.

**Model construction**: This involves the creation of a model that will use the embedded vectors from vectorization to do the specific task it has been designed for.

1. Approach

In this work, we use the IMDB Large Movie Review dataset for training. We begin by applying a standardized data preprocessing pipeline to clean the text reviews. This includes removing unnecessary elements such as URLs, HTML tags, punctuation, and stop words[[2]](#footnote-2). The same preprocessing strategy is applied consistently across all models to ensure comparability. We then experiment with various models, each using different tokenization and vectorization techniques, placing particular emphasis on how these choices affect performance. After training, we evaluate the models on the IMDB test set to assess their effectiveness on the domain they were trained for (movie reviews). To further test the robustness and generalizability of our models, we evaluate them on the Amazon Reviews dataset, which contains user movies comments different from the original IMDB. This helps us understand how well our models transfer to a different but related domain.

* 1. Datasets

In this work, we use two different datasets: the IMDB Large Movie Review Dataset (Andrew Y. Ng et al., 2011) and the Amazon Reviews Dataset and particularly the Movies and TV dataset.

The Large Movie Review Dataset is a collection of 50,000 movie reviews for training a sentiment analysis model. The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg). It also includes an additional 50,000 unlabeled documents for unsupervised learning. This dataset is well documented and serves as a benchmark in many NLP application fields. In this work, this dataset will serve as our benchmark for the evaluation of our models.

The Amazon Review Dataset (McAuley et al, 2024) on the other hand is a large corpus of Amazon reviews regarding their articles. The dataset is available on HuggingFace and is divided into individual parts regarding the type of the articles. In this work, we use the Movie and TV dataset which contains more than 17M reviews on Amazon Movies and subset it into 3 different files (containing 10000 reviews each) which we evaluate separately.

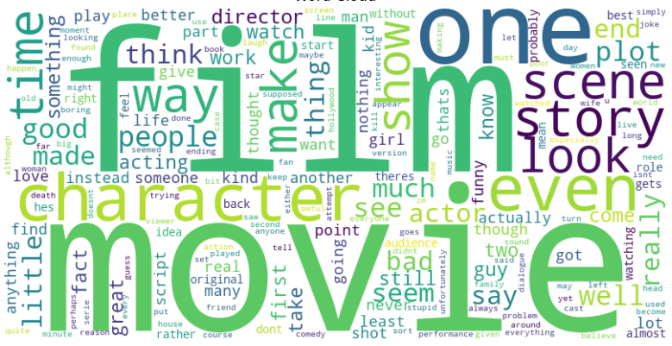
* 1. Data Preparation and Visualization

In order to perform the task of classification more accurately, we first want to take a deep dive into the dataset and get a fundamental understanding of what is inside of it. In doing that, we will grasp what intuitively differentiates a positive comment from a negative one. We visualize the IMDB train dataset.

**Data Preparation:** Preparing the text corpus involves cleaning the text comments from what does not bring information. For that, we removed URLs, HTML Tags, punctuations and stop words

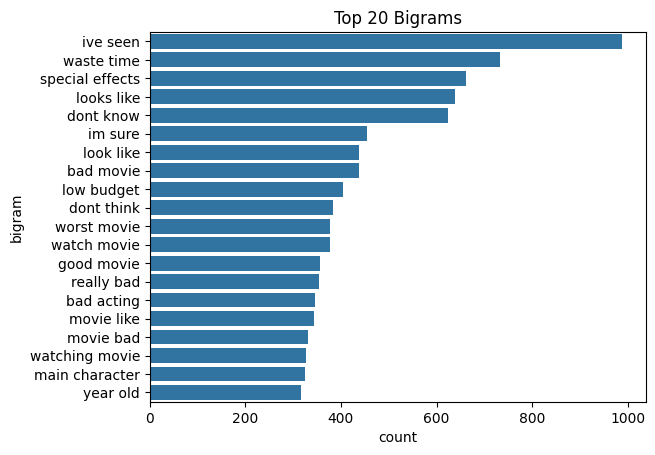
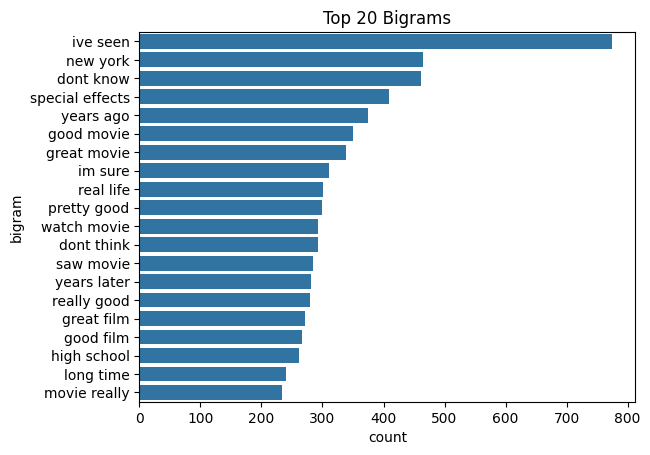
**Data Visualization:** Our visualizations were done on the train set. The train set is composed of labeled movie reviews. There are two balanced classes: positive (1, 12500 reviews) and negative (0, 12500 reviews) in the train set. For each class, we visualized a word cloud[[3]](#footnote-3) and most frequent bigrams[[4]](#footnote-4).

Fig. 1 Word Cloud Visualization (Positive on the left, Negative on the right)



After cleaning our dataset and removing stopwords, what appears clear to the eye is that words like movie, film are quite frequent in our dataset, not to mention that this remark was predictable. But looking carefully, we figured out words that have a good or positive connotation are used in both types of reviews. This suggests the use of negative form for negative comments and/or words like bad, worse etc. in our dataset. This led us to look at the top bigrams. The analysis of top bigrams shows that indeed the combination of two or more words can significantly differentiate the two classes.

Fig. 2 Top 20 Bigrams (Positive on the left, Negative on the right)



* 1. Data Preprocessing

Data Preprocessing is the process of giving a specific form to our data in order to make it suitable for the model we plan to use. In Natural Language Processing, this part is essential since the inputs are texts, we need to find a numerical representation for those texts. Texts are composed of sentences and sentences of words, words of characters. Characters are the base elements of human language but they don't hold meaning on their own. Words on the other hand hold meaning (solely or even depending on the context they are used in). In order to give a numerical representation to texts, we need to divide them into smaller elements which can be used in any sentence. This process in called **tokenization**. We therefore distinguish three types of tokenization:

* **Word based tokenization**: The text is broken down into the words composing it. This makes it so that all words that are in the dataset will be included in the vocabulary (by default). This can lead to huge vocabulary size but generally holds better understanding of each word. Another of its drawbacks is the fact that a word and its plural form are considered different words for example. In our study, for deep learning models, we used a vocabulary size of 20000 to capture most words. Words that are not in the vocabulary are considered **out of vocabulary words** (OOV) and are encoded with the special token **<OOV>**.
* **Character based tokenization**: The text is broken down into the characters (alphabet). This minimize the vocabulary size to only the characters that are in the alphabet but generally fails to grasp the meaning of words. We do not explore the use of character level tokenization for its poor performances.
* **Subword based tokenization** (somewhere in the middle): More complex but it can grasp the concept of root word, prefix and suffix pretty well. Generally used in text generation algorithms, it leads to smaller vocabulary size than word based yet bigger than character based and grasp meaning of words better. The vocabulary size was set to 5000 tokens which is significantly less than the word level tokenization vocabulary size because there is no need for a lot of tokens to grasp many words in this case.

In addition to that, there should be a fixed length of tokens entering our models to stay consistent with each input. Thus, we need to define a maximum length (or context length). Sequences longer than that will be truncated and sequences shorter will be padded with zeros (shifted right). We define our context length to be 500.

After tokenization, we need to encode our tokens into a high dimensional space to give them meaning. We distinguish multiple methods for doing that. For supervised models, we trained a neural network responsible for embedding vectors in a high dimensional space during the training optimization. For unsupervised models, we explore methods such as TF-IDF, Bag of Word and Word2Vec (which is essentially a neural network too).

* 1. Models

We trained multiple models for the task to be performed and paid attention to explore the consequences of training with different preprocessing techniques on our evaluation metrics. In supervised learning, we explore Neural Network Models: Long Short-Term Memory, Gated Recurrent Unit and Transformer Architecture. Unsupervised learning uses methods such as K-Means, HDBScan, etc.

|  |  |  |
| --- | --- | --- |
| **Models (core)** | **Tokenization Method** | **Vectorization method** |
| Supervised Learning | | |
| LSTM, GRU[[5]](#footnote-5)(64 units) | Word level  (MAX\_VOCAB = 20,000, MAX\_LEN = 500) | Embedding (Neural Network)  (OUTPUT\_DIM = 128) |
| LSTM, GRU (64 units) | Subword level Byte Pair Encoder  (MAX\_VOCAB = 5,000, MAX\_LEN = 500) | Embedding (Neural Network)  (OUTPUT\_DIM = 512) |
| LSTM, GRU (64 units) | Subword level (GPT-2 Byte Pair Encoding Tokenizer)  (MAX\_VOCAB = 50,237, MAX\_LEN = 500) | Embedding (Neural Network)  (OUTPUT\_DIM = 512) |
| Transformer (Encoder Only, 2 layers, 4 heads) | Subword level Byte Pair Encoder  (MAX\_VOCAB = 5,000, MAX\_LEN = 500) | Embedding (Neural Network)  (OUTPUT\_DIM = 64) |
| Unsupervised Learning | | |
| KMeans (2 clusters) | Word Level | TF-IDF |
| KMeans (2 clusters) | Word Level | Bag of Words |
| KMeans (2 clusters) | Word Level | Word2Vec |
| HDBScan | Word Level | TF-IDF |
| HDBScan | Word Level | Bag of Words |
| HDBScan | Word Level | Word2Vec |

MAX\_VOCAB determines the size of the vocabulary to reach while training the tokenizer. MAX\_LEN is the context length i.e., the number of tokens the model can process at one iteration. OUTPUT\_DIM is the dimension of the embedding space. For example, OUTPUT\_DIM = 512 means that each token in the vocabulary will be embedded in a 512 dimensional-space thus its vector shape will be (512,).

1. Results

Through our models, we get meaningful results. Although these are not the best results available on the benchmark, our models allow us to derive insights on the importance of preprocessing operations in the making of AI models, in this case, classifiers.

|  |  |  |
| --- | --- | --- |
| Models | Performances | |
| IMDB | Amazon |
| Supervised Learning: Evaluation metric is Accuracy | | |
| WT LSTM | 87% | 84% |
| BPE LSTM | 86% | 88% |
| GPT2 LSTM | 86% | 85% |
| WT GRU | 87% | 88% |
| BPE GRU | 86% | 89% |
| GPT2 GRU | 86% | 78% |
| BPE TRANSFORMER | 87% | 88% |
| GPT2 TRANSFORMER | 88% | 79% |
| Unsupervised Learning: Evaluation metric is Silhouette Score | | |
| TF-IDF KMEANS | 0.1635 | - |
| BOW KMEANS | 0.5394 | - |
| W2V KMEANS | 0.153 | - |
| TF-IDF HDBSCAN | 0.13 | - |
| BOW HDBSCAN | -0.365 | - |
| W2V HDBSCAN | 0.083 | - |

1. Just as in human language, vocabulary here refers to all the tokens the model knows. We often get it by tokenizing the entire corpus of text and selecting the unique tokens. [↑](#footnote-ref-1)
2. A stop word is a word that does not provide any additional information to the sequence for the task we want to perform. Common examples are pronouns, articles, etc., in general words that appears commonly in human language whatever the topic we speak about. [↑](#footnote-ref-2)
3. A word cloud is a visual representation of words that appear in a given text, where the size of each word corresponds to its frequency. The more often a word appears, the larger it will be in the cloud, making it easier to see the most common and important words. [↑](#footnote-ref-3)
4. A bigram is a pair of consecutive words in the sequence. [↑](#footnote-ref-4)
5. This means for the same tokenizer and vectorization method, we have two models: one that uses LSTM as its core and the other that uses GRU instead. [↑](#footnote-ref-5)