Text Representation and Classification using TF-IDF, Faxttext and Naives Bayes Classifier

Ndjebayi Damaris
dsndjebayi@aimsammi.org

Abduljaleel Adejumo
aadejumo@aimsammi.org

Ndeye Ngone Gueye
nngueye@aimsammi.org

Abstract

Text classification is one of the important task today, regarding the automatism in differents fields of knowledge. So, one important thing to take into account is the process of classify automatically and more accurately as possible the document.The concern of this document is to show how well the documents are classify when we use TF-IDF or Fast Text like a machine representation. At the end of the experimentation, we found that byusing Bernouilli Naives Bayes Classifier with Laplace smoothing, TF-IDF perform well than Fasttext in the task of classification.

Keywords: TF-IDF, FastText, Naive Bayes

1 Introduction

The task of classifying text documents is nowadays a very important one as technology evolves to generate a large number of text documents to be categorized to meet a specific need. The main objective of the classifier is to obtain a set of characteristics that remain relatively constant for distinct categories of text documents and to classify the documents with other related text documents (Chandran, 2021). The general steps in text classification include word segmentation and vocabulary building, feature input selection, text representation, training model, and pre- diction category. Feature selection is one of the crucial steps in text classification; indeed, one wants to be able to select features that best express the content of the text and the differ- ence between different texts for classification (Steven Bird and Loper, 2019). In this report we present first the textual representation methods and explain how the work; by focusing on TF-IDF and Fast text in a hand, and after, we analyze how these two approaches influence text classification using Naive Bayes classifier.

2 Text Representation

Text representation is one of the fundamental problems in text mining and Information Retrieval (IR). It aims to numerically represent the unstructured text documents to make them mathematically computable (Yan et al.,). For a given set of text documents $D = \{d_i\}_{i=1}^{n}$, where each d_i stands for a document, the problem of text representation is to represent each d_i of D as a point s_i in a numerical space S, where the distance/similarity between each pair of points in space S is well defined (Kowsari et al., 2019). We have two differents types of text representations : Discrete representation and continuous representation.

2.1 Discrete Representation

These are representations where words are represented by their corresponding indexes to their position in a dictionary from a larger corpus (Nigam et al.,). There are serveral methods of discrete representation, such as One-Hot encoding, Bag-of-words representation (BOW), Basic BOW — CountVectorizer, Advanced BOW — TF-IDF. In this report, we are going to talk more about TF-IDF method.

2.1.1 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: one which gives us information on how often a term appears in a document (Term Frequency), and another gives us information about the relative rarity of a term in the collection of documents (Inverse Document Frequency).

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

→ **Term Frequency**

Term frequency works by looking at the frequency of a particular term you are concerned with relative to the document. There are multiple measures, or ways, of defining frequency (One, 2011):

- Number of times the word appears in a document (raw count).
- Term frequency adjusted for the length of the document (raw count of occurences divided by number of words in the document): It is the most communly used method.

$$TF(t, d) = \frac{F_{t,d}}{\sum_{t' \in d} F_{t',d}}$$

- Logarithmically scaled frequency (e.g. log(1 + raw count)).
- Boolean frequency (e.g. 1 if the term occurs, or 0 if the term does not occur, in the document).

→ **Inverse Document Frequency**

Inverse document frequency looks at how common (or uncommon) a word is amongst the corpus. IDF is computed as follows where **t** is the term (word) we are looking to measure the commonness of and **N** is the number of documents (**d**) in the corpus (**D**). The denominator is simply the number of documents in which the term, t, appears in.

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

It can be possible for a term to not appear in the corpus at all, which can result in a divide-by-zero error. One way to handle this is to take the existing count and add 1. Thus making the denominator (1 + count); so the previous formula is usually implemented like the following one:

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}| + 1}$$

Now, we have seen how text is represented using TF-IDF method; This method helps to reduce the drawback faced with the method like **Bag of words**[1], which creates a set of vectors containing the count of word occurrences in the document, while the TF-IDF model contains information on the more important words and the less important ones as well. However, there are certain disadvantages – Semantics and Context are not captured at all i.e. the meaning is not modeled effectively in the above methods.

To push the limits, we need better, stronger and more robust representation of corpus. These are called **embeddings**[2], which take into account the context of a word to give it representation. The next section will talk more about.

2.2 Continuous or distributed Representation

The vector representation of words is effectively accomplished by word embeddings. A word embedding is an approach for representing words in vector form, so it is a function that maps each word type to a single vector. It is the process of converting words into dense vectors and these vectors have much lower dimensionality than the size of the vocabulary. It provides similar vector representations for words that have similar meanings. It helps the model to capture the linguistic meaning of the word. Some popular word embedding techniques are Word2Vec, GloVe, FastText, ELMo. In the following subsections , we will talk about Word2Vec and FastText.

2.2.1 Word2Vec

Word2Vec is a word embedding technique to represent words in vector form. It takes a whole corpus of words and provides embedding for those words in high-dimensional space. Semantic and syntactic links between words are also maintained by the Word2Vec model. The Word2Vec model is used to determine how closely linked words are across the model. The word2vec model computes the vectors using two main architectures: CBOW and Skip-gram.

→ CBOW works

CBOW, ContinuousBag- of-Words, in this method, the context is given, and the target word is predicted. If a sentence is given and a word is missing, the model must predict the missing word. The CBOW model architecture tries to predict the current target word (the center word) based on the source context words (surrounding words).
Considering a simple sentence, " the dog behind the car is black", this can be pairs of (context_window, target_word) where if we consider a context window of size 2, we have examples like ([dog, the], behind), ([the, behind], dog), ([car, black], is) and so on. Thus the model tries to predict the target_word based on the context_window words (Nussbaum, 2018).

The Word2Vec family of models are unsupervised, which simply means that you may give them a corpus without giving them any labels or other information, and they will be able to build dense word embeddings from the corpus. But to get these embeddings after you have this corpus, you will still need to use a supervised classification algorithm. But we'll achieve that directly from the corpus, without using any supplementary data. By using the context words as our input, X, and trying to predict the target word, Y, we can now model this CBOW architecture as a deep learning classification model.

→ Skip-gram works

Skip-gram, in this method, the target word is given, and the probability of the context word is predicted (Sarkar, 2018).

The Skip-gram model architecture usually tries to achieve the reverse of what the CBOW model does. It tries to predict the source context words (surrounding words) given a target word (the center word). Considering our simple sentence , " the dog behind the car is black".The skip-gram model's aim is to predict the context from the target word, the model typically inverts the contexts and targets, and tries to predict each context word from its target word. Hence the task becomes to predict the context [dog, the] given target word 'behind' or [car, black] given target word 'is' and so on. Thus the model tries to predict the context_window words based on the target_word. The goal is to independently predict the presence (or absence) of context words. For the word at position t we consider all context words as positive examples and sample negatives at random from the dictionary. For a chosen context position c, using the binary logistic loss, the skip-gram objective function sums the log probabilities of the surrounding words to the left and right of the target word w_t and the log probabilities of sample negatives to produce the following objective (Mikolov et al., 2013):

$$\sum_{t=1}^{T} \Big[\sum_{c \in C_t} l(s(w_t, w_c)) + \sum_{n \in N_{t,c}} l(-s(w_t, n)) \Big]$$

Where:

- $l(x) = log(1 + e^{-x})$: the logistic loss fonction,

- $s(w_t, w_c) = u_{w_t}^T v_{w_c}$: the scoring function (vectors $u_{w_t}^T$ and v_{w_c} , corresponding respectivelywords w_t and w_c),

- $N_{t,c}$ is a set of negative examples sampled from the vocabulary.

So, we feed our skip-gram model pairs of (X, Y) where X is our input and Y is our label. We do this by using [(target, context), 1] pairs as positive input samples where target is

[1]a representation of text that describes the occurrence of words within a document

[2]a relatively low-dimensional space into which you can translate high-dimensional vectors.

our word of interest and context is a context word occurring near the target word and the positive label 1 indicates this is a contextually relevant pair. We also feed in [(target, random), 0] pairs as negative input samples where target is again our word of interest but random is just a randomly selected word from our vocabulary which has no context or association with our target word. Hence the negative label 0 indicates this is a contextually irrelevant pair. We do this so that the model can then learn which pairs of words are contextually relevant and which are not and generate similar embeddings for semantically similar words.

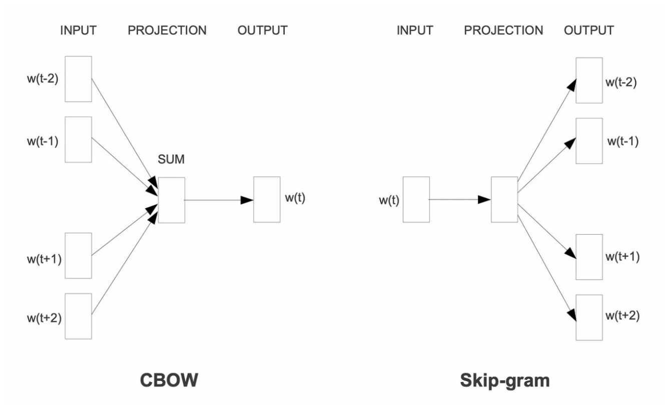

figure :CBOW and Skip-gram models architectures

→ **Limitation of Word2Vec**

- Word2Vec cannot provide embeddings for out-of-vocabulary words, can't provide morphologically rich languages;
- It cannot provide embeddings for out-of-vocabulary (OOV) words[3],
- Assigning a distinct vector to each word.

→ **Character n-gram**

N-gram[4] language models are based on probabilities of chunks of word.

Example:"the student opened their books"

• unigram: unit of single word – "the", "student", "opened", "their" ,"books"

• bigrams: unit of double words – "the student", "student opened", "opened their", "their books"
•trigram:unit of triple words –"the student opened","student opened their"

• 4-gram: unit of 4 words – "the student opened their" • The main idea behind n-gram models is to collect statistics about the frequency of different n-grams and use this to predict the next word.

With character n-gram each word w is represented as a collection of n-grams. At the start and end of words, we add special boundary symbols (" $\langle$"*and*"$\rangle$ ") that make it possible to distinguish prefixes and suffixes from other character sequences.

[3]OOV are words that do not occur while training the data and are not present in the model's vocabulary

[4]An n-gram is a chunk of n consecutive words.

For example: fasttext will have character n-gram $\langle$fa , fas , ast , stt , tte , tex , ext , xt$\rangle$ with n=3

FastText include the word w itself in the set of its n-grams, to learn a representation for each word (in addition to character n-grams) i.e:

$\langle$fa, fas, ast, stt, tte, tex, ext, xt$\rangle$and $\langle$fasttext$\rangle$

2.3 FastText

Word embedding techniques like word2vec and GloVe provide distinct vector representations for the words in the vocabulary. This leads to ignorance of the internal structure of the language. This is a limitation for morphologically rich language as it ignores the syntactic relation of the words. As many word formations follow the rules in morphologically rich languages, it is possible to improve vector representations for these languages by using character-level information.

To improve vector representation for morphologically rich language, FastText provides embeddings for character n-grams, representing words as the sum of these embeddings. So FastText works with subwords model and propose a different scoring function s, in order to take into account this information(morphologically rich language, on vector each word).

Assuming we are given a dictionary of n- grams of size G. Given a word w, let us denote by $S_w\{1, ..., G\}$ the set of n-grams appearing in w. We associate a vector representation z_g to each n-gram g. We represent a word by the sum of the vector representations of its n-grams. We thus obtain the scoring function (Bojanowski et al., 2016):

$$s(w_t, w_c) = \sum_{g \in S_w} z_g^T v_c$$

It is an extension of the word2vec model. Word2Vec model provides embedding to the words, whereas fastText provides embeddings to the character n-grams. Like the word2vec model, fastText uses CBOW and Skip-gram to compute the vectors.

FastText can also handle out-of-vocabulary words, i.e., the fast text can find the word embeddings that are not present at the time of training.

Out-of-vocabulary (OOV) words are words that do not occur while training the data and are not present in the model's vocabulary. Word embedding models like word2vec or GloVe cannot provide embeddings for the OOV words because they provide embeddings for words; hence, if a new word occurs, it cannot provide embedding.

Since FastText provides embeddings for character n-grams, it can provide embeddings for OOV words. If an OOV word occurs, then fastText provides embedding for that word by embedding its character n-gram.

3 Naive Bayes

Naive Bayes is a probabilistic machine learning algorithm that can be used in a wide variety of classification tasks. Based on the Bayes theorem, naïve Bayes classifiers make the assumption that each feature in a dataset is independent of the others. Although the independence assumption is sometimes broken in practice, Naive Bayes nevertheless frequently delivers excellent results in the disciplines of text/document categorization. Spam filtering (which classifies a text message as spam or not-spam) and sentiment analysis (which classifies a text message as positive or negative review) are common applications (Webb, 2021).

By adopting the naive assumption that each feature (for example, each column in an input vector x) is statistically independent of all the other features, the NB algorithms simplify the conditional probability distribution. This makes the algorithm great since the naive assumption boosts the algorithm's capacity to parallelize. Furthermore, the general technique of estimating simple co-occurrence probabilities between features and class labels allows the model to be trained progressively, allowing it to support datasets that are too large to fit into memory.

3.1 Bayes Theorem

Bayes' Theorem is a mathematical formula that calculates conditional probabilities, which measure the likelihood of an event happening based on the occurrence of another event. It takes into account assumptions, presumptions, assertions, or evidence about the occurrence of the first event(Chen, 2021).

The formula is given:-

$$P(A \mid B)=\frac{P(A)\, P(B \mid A)}{P(B)}$$

$P(A \mid B)$: Is the probability of A occurring giving B as occurred. This is called the posterior

$P(B \mid A)$ Is the probability of B occurring giving A as occurred. This is called the likelihood

$P(A)$: Is the probability of A occurring. This is called the prior

$P(B)$: Is the probability of B occurring. This is called the marginal probaility

"In simpler terms, Bayes' Theorem is a way of finding a probability when we know certain other probabilities."

3.2 Naive Bayes assumptions

1. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

2. Equal contribution the outcome.

3.3 Method

Giving a set of feature $(x_1, x_2, x_3, ..., x_n)$ are element of X and with label y the bayes theorem can be written as:

$$P(y \mid X)=\frac{P(y)\, P(X \mid y)}{P(y)}$$

The variable y is the class variable, which represents if the y occurred or not given the conditions. Variable X represents the parameters/features.

By substituting for X and expanding using the chain rule we get,

$$P(y \mid X)=\frac{P(y)\, P(x_1 \mid y)\, P(x_2 \mid y)\, P(x_3 \mid y) \ldots P(x_n \mid y)}{P(y)}$$

Now, you can easily determine the values for each variable by examining the dataset and plugging them into the equation. Since the denominator in the equation remains constant for all entries in the dataset, it can be eliminated, and proportionality can be introduced for simpler calculations.

$$P(y/x_1, , x_n) \propto P(y) \prod_{i=1}^{n} P(x_i/y)$$

In our scenario, the class variable y has only two possible outcomes, yes or no. However, in other cases, the classification may involve multiple variables. Hence, our goal is to determine the class variable y with the highest probability in order to make the most accurate prediction.

$$y = \underset{y}{\operatorname{argmax}} P(y) \prod_{i=1}^{n} P(x_i/y)$$

By utilizing the function mentioned above, we can derive the class based on the given predictors or features (Shimodaira, 2015).

The posterior probability $P(y|X)$ can be computed by following these steps: first, create a Frequency Table for each attribute in relation to the target. Then, transform the frequency tables into Likelihood Tables. Finally, employ the Naïve Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is considered as the prediction outcome. Presented below are the Frequency and Likelihood Tables for all three predictors.

3.4 Problem: estimating the joint PD or CPD isn't practical

- Severely overfits: Naïve Bayes may encounter the issue of zero probability when a class label and attribute value do not occur together in the training data. One approach to overcome this is to apply Laplace smoothing, which adds a count of one to attribute value-class combinations that are not observed, preventing zero probabilities and improving classifier accuracy.

3.5 Laplace Smoothing

Notice that some probabilities estimated by counting might be zero

- Possible overfitting!
- Fix by using Laplace smoothing:
- Adds 1 to each count

If a given token and class never occur together in the training data, then the probability estimate for that token conditioned on that class will be zero. Recall, the probability estimate is directly proportional to the number of times a given token occurs.

$$P\left(X_j = v \mid Y = y_k\right) = \frac{\mathrm{c}_v + 1}{\sum_{v' \in \text{values}(X_j)} \mathrm{c}_{v'} + \mid \text{ values } \left(X_j\right) \mid}$$

where

$-c_v$ is the count of training instances with a value of v for attribute j and class label y_k

- $\left|\text{values}\left(X_j\right)\right|$ is the number of values X_j can take on

3.6 Type of Naïve Bayes classifiers

Naïve Bayes classifiers can be categorized into three types:

Bernoulli Naïve Bayes Classifier: This model is also used for document classification, but it specifically handles binary variables or boolean features. It is suitable for tasks where binary term occurrence (e.g. presence or absence of a word in a document) is used instead of term frequencies.

Multinomial Naïve Bayes Classifier: This model is commonly used for document classification tasks, where feature vectors represent the frequencies of events generated by a multinomial distribution. It is suitable for handling discrete features.

Gaussian Naïve Bayes Classifier: This model assumes that continuous feature values follow a Gaussian distribution (Normal distribution) and is suitable for handling continuous features. The distribution of feature values forms a bell-shaped curve that is symmetric around the mean.

4 Implementation

For the implementation, we are going to build a naive bayes classifier and compare the results for both TF-IDF and Fast text representation method; by following the steps descibe on the figure 4.3:

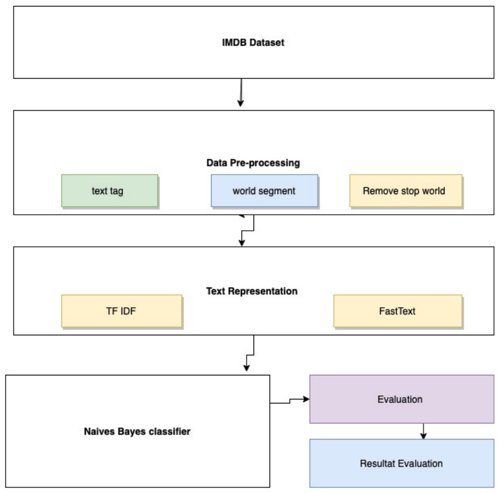

figure :Implementation process

4.1 Dataset

We used IMDB dataset http://ai.stanford.edu/ amaas/data/sentiment/link is a popular dataset for text and language-related machine learning tutorials. It contains 50,000 movie reviews (25,000 in training and 25,000 in testing) from IMDB, as well as each movie review's binary sentiment: positive or negative.

4.2 Preprocessing and Implementation

For the preprocessing, we used regular expressions to :

- Remove the redundant character
- Remove html tags
- reduce all the upper characters to lower characters
- replace extra caracter with space
- Remove URLs
- Remove punctuations
- remove stopwords
- remove emojis

We encoded also sentiment columns using one hot encoding and we train first train from scratch TF-IDF, due to the lack of ressources, we switched on sklearn version of TF-IDF, and used it to train the Naive Bayes model. We implement the Fastext from gensim.

4.3 Results

The results find by applying Naive Bayes classifier on TF-IDF and fastext representation are stored in the following table

Representation method	Accuracy	SKlearn accuracy
TF-IDF	86.165	85.165
FastText	50.075	52.205

Table 1: Results of implementation

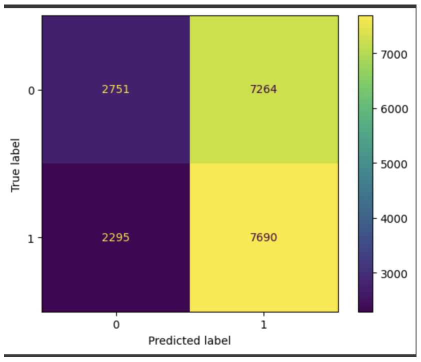

figure :Confusion matrix for FastText with Naive Bayes Classifier

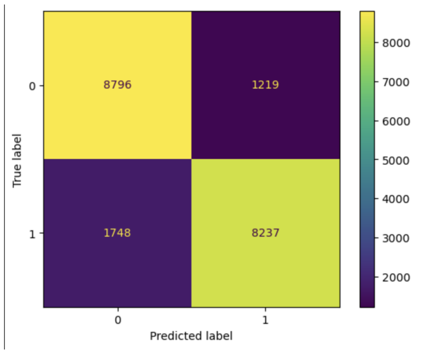

figure :Confusion matrix for tf-idf with Naive Bayes Classifier

5 Discussion/Conclusions

Based on the previous results, we notice that TF-IDF representation perform better than FastText reprensentation in the task of Classification using Naive Bayes Classifier with laplace smoothing; This result may be explain with the fact that laplace smoothing use like a counting measure to classify documents and TF-IDF as a discrete type of text representation, can perform easily to this, comparing to Fasttext representation which is more context word oriented, and also a continuous method representation in text representation.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Sundaresh Chandran. 2021. Introduction to text representations for language processing — part 1.

Hu S. Hua R. et al Chen, H. 2021. Improved naive bayes classification algorithm for traffic risk management.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey.

Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. pages 1–9, 01.

Aastha Nigam, Pingjie Tang, Henry K. Dambanemuya, and Bryan (Ning) Xia. Text representation for machine learning applications.

Barry D. Nussbaum. 2018. Statistics: Essential now more than ever. *Journal of the American Statistical Association*, 113(522):489–493.

Capital One. 2011. Understanding tf-idf for machine learning.

Dipanjan Sarkar. 2018. Implementing deep learning methods and feature engineering for text data: The skip-gram model.

Hiroshi Shimodaira. 2015. Text classification using naive bayes.

Ewan Klein Steven Bird and Edward Loper. 2019. *Natural Language Processing with Python - chapter 6: Learning to Classify Text*.

Webb. 2021. Encyclopedia of machine learning. springer.

Jun Yan, LING LIU, and M. TAMER ÖZSU. *Encyclopedia of Database Systems*.