Sentiment Analysis using Supervised Machine Learning

Anant Mahajan, Anshuman Ray, Ashish Verma,
Shreya Kohad, Prasheel N Thakare

Department of Electronics and Communication Engineering
Shri Ramdeobaba College of Engineering and Management, Nagpur

ABSTRACT

This paper acquaints a methodology with assumption investigation which utilizes different content standardization methods in Natural Language Processing (NLP) for converting a text into vector and briefly explains the the importance of standardization methods and how they are used in python with the help of its Natural Language Toolkit (NLTK) library. Finally this paper analyzes different algorithms in Supervised Machine Learning with comparison of two Machine Learning models, i.e., Logistic Regression and Naive Bayes for linear classification with the help of a data set.

Keywords: *Natural Language Processing; Text Normalization; Supervised Machine Learning; Logistic Regression; Naive Bayes.*

1 INTRODUCTION

Film surveys help clients to choose if the film merits their time. A rundown of all surveys for a film can assist clients with settling on this choice by not burning through their time perusing all audits. Film rating sites are regularly utilized by pundits to post remarks and rate motion pictures which assist watchers with choosing if the film merits viewing. Supposition investigation can decide the demeanor of pundits relying upon their surveys. Assessment investigation of a film audit can rate how positive or negative a film survey is and subsequently the general rating for a film. Accordingly, the way toward comprehension if an audit is positive or negative can be computerized as the machine learns through preparing and testing the information [1].

Characteristic language preparing (NLP) is the connection among PCs and human language.Normal language alludes to discourse investigation in both discernible discourse, just as text of a language. NLP systems capture meaning from an input of words (sentences, paragraphs, pages, etc.). This project intends to execute different content handling strategies in NLP and afterward manufacture a Machine Learning Model so as to order the given survey as positive or negative.

2 TEXT NORMALIZATION

The process of transforming a text into a canonical (standard) form is called as Text Normalization. A few stages must be acted so as to standardize the content and convert it into fitting structure as we can't give the PC text as information ,which would then be able to be given as contribution to the machine learning (ML) model. Thus the amount of different information that the computer has to deal with gets reduced subsequently and improves the efficiency. Library utilized for this is given in [2]. Steps associated with this cycle are shown in Figure 1 and explained in the following sections.

2.1 Removing Stopwords

A famous methodology to diminish the commotion of literary information is to eliminate stopwords by utilizing precompiled stopword records or more advanced strategies for dynamic stopword distinguishing proof[3]. A stopword is an ordinarily utilized word, (for example, "the", "an", "an", "in") that a web crawler has been modified to disregard, both when ordering sections for looking and while recovering them as a result of a search query. Natural Language Toolkit (NLTK) in python has a rundown of stopwords put away in 16 unique dialects. Such words have no commitment to the conclusion of a specific sentence and henceforth can be deleted from the first content. Consider this content string – "There is a pen on the table". Presently, the words 'is', 'an', 'on', and 'the' add no importance to the announcement while parsing it. While words like 'there', 'book', and 'table' are the catch phrases and mention to us what the sentence is about. However, we should avoid removing stopwords when performing the tasks in which output speech is more significant.

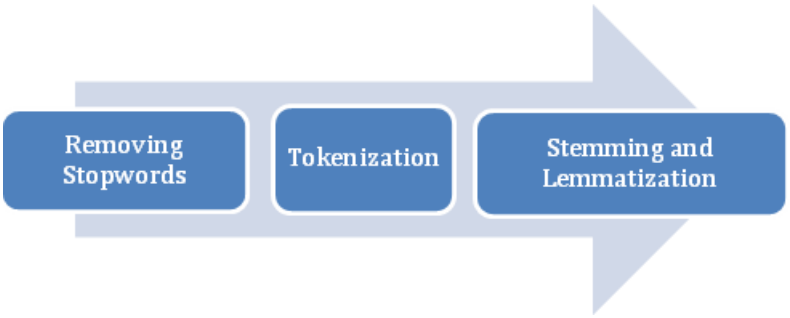


Figure 1: Text Normalization

2.2 Tokenization

Tokenization is a method of isolating a bit of text into smaller units called tokens. Here, tokens can be either words, subwords, or characters. Subsequently, tokenization can be comprehensively characterized into 3 kinds – word, character, and subword tokenization. For instance, think about the sentence: ”Never surrender”. The most well-known method of framing tokens depends on space [4]. Expecting space as a delimiter, the tokenization of the sentence brings about 2 tokens – Never-surrender. As every token is a word, it turns into a case of word tokenization. Thus, tokens can be either characters or subwords as shown in this Example 1.

Example 1. *Let us consider “smarter”. Then Character tokens: s-m-a-r-t-e-r and Subword tokens: smart-er.*

2.2.1 Different types of tokenizers in NLTK

Tokens can be paragraphs, sentences, or individual words. NLTK’s tokenize module gives a bunch of tokenizers to part the content into tokens. Here, a few of them are listed [5].

TweetTokenizer: It is designed to be flexible and easy to adapt to new domains and tasks. This can reduce length if repeated more than 3 times and removes the userhandle. Here is an example:

```
>>>tknzr = TweetTokenizer(strip_handles=True, reduce_len=True)
>>>s1 = ’ adam: This is waaaaayyyy too much for you!!!!!!’
>>>tknzr.tokenize(s1)
Tokens: [’:’, ’This’, ’is’, ’waaayyy’, ’too’, ’much’, ’for’, ’you’, ’!’, ’!’, ’!’]
```

SExprTokenizer: This tokenizer is Symbolic Expressions Tokenizer. It splits a string into substrings using a regular expression which matches either the tokens or the separators between tokens. It isolates the string into tokens dependent on parenthesized articulations and whitespace. Here is an example:

```
>>>SExprTokenizer().tokenize(’(a b (c d)) e f (g)’)
Tokens: [’(a b (c d))’, ’e’, ’f’, ’(g)’]
```

SpaceTokenizer: Based on space, tokens are created. Here is an example:

```
>>>s1 = ’@remy: This is waaaaayyyy too much for you!!!!!!’
>>>tknzr.tokenize(s1)
Tokens: [’@remy: ’, ’This’, ’is’, ’waaaaayyyy ’, ’too’, ’much’, ’for’, ’you!!!!!!’]
```

RegexpTokenizer: This tokenizer splits a string into substrings using a regular expression which matches either the tokens or the separators between tokens. Parameter of pattern is used to build this tokenizer. This can be an ideal tokenizer in classification tasks like sentiment analysis, since, more flexible and more control is in our hands to decide how to form tokens. A regular expression (sometimes called a rational expression) is a sequence of characters that define a search pattern, mainly for use in pattern matching with strings, or string matching, i.e., ”find and replace” - like operations [6]. Regular expression which can be used: ’ w +’, where w matches any word character. Basically alpha-numeric, special characters are excluded like , !, %, $. The customary articulation ab+c will give abc, abbc, abbc,., etc. For example

Consider the review : Movie was awesome!. I would love to watch it 100 times and it costed me 50$ for one show!
For this sentence after applying RegexpTokenizer tokens generated are
Tokens : [’Movie’, ’was’, ’awesome’, ’I’, ’would’, ’love’, ’to’, ’watch’, ’it’, ’100’, ’times’, ’it’, ’costed’, ’me’, ’50’, ’for’, ’one’, ’show’]

2.3 Deriving Root Word

In the zones of Natural Language Processing we run over circumstances where at least two words have a typical root. For instance, the three words - ’concurred’, ’concurring’ and ’concurrable’ have a similar root word concur. A search including any of these words should regard them as a similar word which is the root word. In NLP, there are two principle procedures to create root words specifically - Stemming and Lemmatization. Stemming and Lemmatization are Text Normalization (also once in a while called Word Normalization) methods in the field of Natural Language Processing that are utilized to get root words from the inflected words [7].

Stemming and Lemmatization both produce the root type of an inflected word. The thing that matters is that stem probably won’t be a genuine word while, lemma is a real language word [8]. Stemming calculation works by cutting the postfix from the word. From a more extensive perspective cuts either the start or end of the word. Consider an example:

Stemming for ”studies” is ’studi’, Stemming for ”studying” is ’studi’, Stemming for ”cries” is ’cri’.

Actually, Lemmatization is an all the more remarkable activity, and it thinks about morphological examination of the words. It restores the lemma which is the base type of all its inflectional structures. Inside and out semantic information is needed to make word references and search for the correct type of the word.

Lemma for ”studies” is ’study’, Lemma for ”studying” is ’study’, Lemma for ”cries” is ’cry’, Lemma for ”cry” is ’cry’.

2.3.1 Types of Stemmers in NLTK

There are two types of stemmers:

1. PorterStemmer
2. LancasterStemmer

There are other non-english stemmers also.

PorterStemmer: In this the calculation doesn’t follow phonetics rather a set of 5 rules for various cases that are applied in stages (bit by bit) to create stems. It is known for its effortlessness and speed. It utilizes Suffix Stripping to create stems [9]. For example:

”Connections” → ’Connect’, ”Connected” → ’Connect’, ”Connecting” → ’Connect’, ”Connection” → ’Connect’

LancasterStemmer: It is an iterative calculation with one table containing around 120 standards listed by the last letter of a suffix. In every cycle, it attempts to locate a material guideline by the last character of the word. Each standard indicates either an erasure or substitution of a completion. In the event with no such guideline, it ends. It additionally ends if a word begins with a vowel and there are just two letters left or if a word begins with a consonant and there are just three characters left. If something else is there, the standard is applied and the cycle rehashes. LancasterStemmer is likewise basic, yet hefty stemming because of emphases, and over-stemming may happen. Over-stemming makes the stems not linguistics, they may have no significance. Over-stemming makes the stems not phonetic, or they may have no meaning. For instance:

”destabilized” is stemmed to ’dest’ in LancasterStemmer while, utilizing PorterStemmer it is ’destabl’

2.4 Feature Extraction

Machine Learning calculations can’t take a shot at the crude content legitimately. Along these lines, we need some component extraction procedures to change over content into a matrix (or vector) of highlights [10]. Probably the most well known strategies that include extraction are:

1. Bag-of-Words
2. TF-IDF

Bag-of-Words: It is a method to extract features from text documents. These highlights can be utilized for preparing ML calculations. It makes a jargon of the apparent multitude of extraordinary words happening in all the archives in the preparation set. In this we make a Feature Matrix based on one hot encoding. A significant disadvantage in utilizing this model is that it leads to a high dimensional feature vector due to large size of vocabulary, *V*. Bag-of-words doesn't leverage co-occurrence statistics between words. It prompts a profoundly scanty vectors as there is nonzero esteem in measurements comparing to words that happen in the sentence. The request for the event of words is lost, as we make a vector of token in randomized order - 'a good movie', 'not a good movie', 'did not like'. One solution for this is considering N-grams (mostly bigrams) instead of individual words, i.e., unigrams.

TF-IDF Vectorizer: TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. Term frequency specifies how frequently a term appears in the entire document. It can be thought of as the probability of finding a word within the document and can be expressed as

$$tf(w_i, r_j) = \frac{No.\text{ of times } w_i \text{ occurs in } r_j}{Total\text{ no. of words in } r_j}$$

A different scheme for calculating *t f* is log normalization and it is formulated as

$$t\,f\,(t, d) = \log(1 + f_{t,d}\,)$$

IDF: IDF stands for Inverse Document Frequency. The inverse document frequency is a measure of whether a term is rare or frequent across the documents in the entire corpus. It highlights those words which occur in very few documents across the corpus, or in simple language, the words that are rare have high IDF scores. Mathematically,

TF-IDF = TF * IDF

Therefore, a high TF-IDF score is obtained by a term that has a high frequency in a document, and low document frequency in the corpus [11].

3 CLASSIFICATION ALGORITHMS IN ML

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the input data and then uses this learning to classify new observations. Few types of classification algorithms in machine learning are:

1. Logistic Regression
2. Naive Bayes

3.1 Logistic Regression

It is a statistical method for analyzing a data set in which there are one or more independent variables that determine an outcome. One should consider utilizing logistic regression when the *Y* variable takes on just two qualities [12]. Such a variable is alluded to as *binary* or *dichotomous*. Dichotomous essentially implies two classifications, for example - yes/no, deficient/non-blemished, achievement/disappointment, etc. Binary refers to 0's and 1's.

3.2 Naive Bayes Classifier

The Naive Bayes classifier is a simple classifier that classifies based on probabilities of events. It is an arrangement strategy dependent on Bayes' Theorem with the supposition of freedom among indicators [13].

$$P(A\backslash B) = \frac{P(B\backslash A)P(A)}{P(B)}$$

where, P(A\B) is the probability of A given B. Here, A will be dependent variable (which is to be predicted) and B will be independent variable (Features).

4 APPLICATION

In this paper we have made a comparison of the Logistic Regression and Naive Bayes methods for sentiment analysis and shown the analysis.

4.1 Logistic Regression Model v/s Naives Bayes Model

There are prominent models to wade in various ML issues. Two of them Logistic regression and Naive Bayes are popularly recognized ,these two provide results which are analogous but still are contrasting,the reason is they take into account different viewpoints. Naive Bayes is based on the concept that it extracts probability of the situation that a component vector is related with a label. It is based on the Bayes's hypothesis. The algorithm have certain assumptions which is unaffected from the realistic facts which says the component vectors are independent which is totally a hypothetical condition and is not always feasible. Whereas Logistic regression is an algorithm which calculates the probability that a feature is associated to a class. This algorithm is preferred because it can give satisfactory results even if the some of the features are found to be associated to a certain class. We have used the following features to do the comparison.

1. Both algorithms (Naive Bayes and Logistic Regression) are used for classification Cases
   Though both algorithms are mainly used for solving problems that include classification tasks , the main difference is , Logistic Regression is limited for only binary classification e.g. Predicting whether a person is infected with a disease or not , mail is spam or not spam , given sentiment is positive or negative , etc. whereas Naive Bayes Algorithm can be applied on multiclass classification problems also.

2. Algorithm's Working
   The workings of both the algorithms are very different . Naive Bayes algorithm is a probabilistic approach whereas , in Logistic Regression we use sigmoid function as activation function to map the result between 0 and 1 . Naive Bayes classification calculation includes P(y—x) (which means probability of y given x) . So when there are multiple features then Naive Bayes classifier assumes that all features are independent of each other which does not hold true while dealing with real world problems . Therefore , if binary classification is considered , then it is generally observed that logistic regression gives better results as compared to Naive bayes classifier .

3. Model assumptions.
   Naive Bayes assumes all the features are different from each other.It is dependent upon each other alot hence accuracy is less. Logistic regression splits it linearly, hence it is important to watch the accuracy according to it.

4. Way to deal with be followed to improve model outcomes.
   Naive Bayes: When the preparation information size is a bit unequal with the data and information on earlier probabilities help in improving the outcomes.

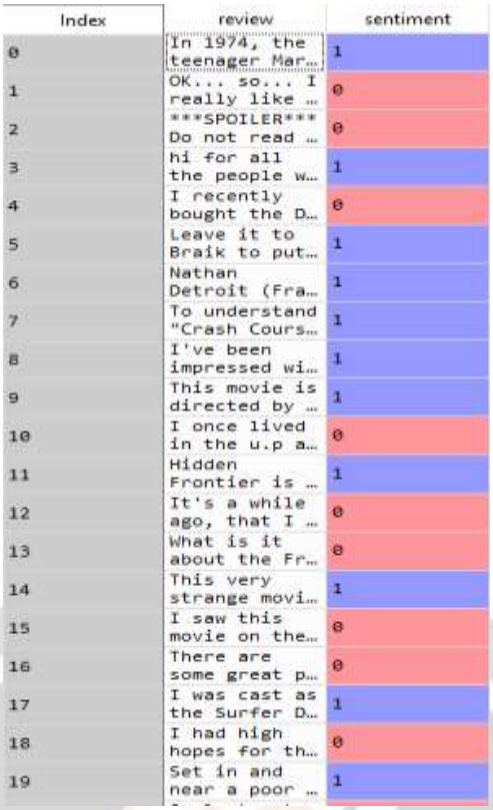
   Logistic regression: When the training data size is small compared to various features which is already supported with it.

4.2 Results

The data shown in Figure 2 of movie reviews was taken from [14] and was implemented in Spyder IDE using python.

This data was trained and tested in two machine learning classification models, i.e., Logistic Regression and Naive Bayes for linearly classifying whether the review is positive or negative. 80% of data was trained and rest 20% was tested to check whether the model was accurate or not. A model is said to be accurate if the output of the model matches with the given sentiment, where '0' means positive and '1' means negative. Hence, accuracy score was calculated for both the ML models respectively. The obtained results are shown in and Figure 3 and Figure 4.

From the results shown in the figures we see that Logistic Regression classification for movie-reviews has achieved the score on training data as 93.59% and on testing data as 89.76%. The classification accuracy score with Naive Bayes show the training data score as 90.6% and testing data score as 86.14%. The findings indicate that Logistic Regression classification for movie-reviews has achieved higher classification accuracy score as compared to Naive Bayes, as the linearly separable and the classification task in this case is binary, i.e. positive or negative. Therefore, logistic regression is found to be more efficient and accurate as compared to Naive Bayes classification algorithm.



Index	review	sentiment
0	In 1974, the teenager Mar…	1
1	OK... so... I really like …	0
2	***SPOILER*** Do not read …	0
3	hi for all the people w…	1
4	I recently bought the D…	0
5	Leave it to Braik to put…	1
6	Nathan Detroit (Fra…	1
7	To understand "Crash Cours…	1
8	I've been impressed wi…	1
9	This movie is directed by …	1
10	I once lived in the u.p a…	0
11	Hidden Frontier is …	1
12	It's a while ago, that I …	0
13	What is it about the Fr…	0
14	This very strange movi…	1
15	I saw this movie on the…	0
16	There are some great p…	0
17	I was cast as the Surfer D…	1
18	I had high hopes for th…	0
19	Set in and near a poor …	1

Figure 2: Data for comparison of the two ML algorithms

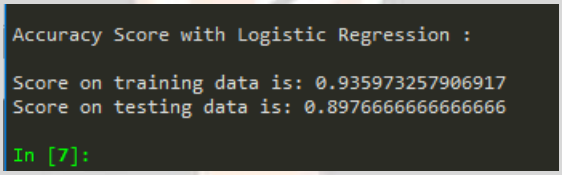


Figure 3: Accuracy score with logistic regression

Figure 4: Accuracy score with Naive Bayes

5 CONCLUSION

In this paper we have reviewed different standardization methods for Natural Language Processing its importance and implementation in python using its NLTK library. This standardization methods are used for normalization of text which can also be called as *data preprocessing* before feeding it into the machine learning models for its linear classification.

Comparison of Logistic Regression and Naive Bayes models for linear classification also have been discussed and implemented for a given dataset of movie-reviews. After a brief comparison, it turned out that Logistic Regression model was more accurate than Naive Bayes model for the given movie-reviews dataset.

References

[1] Igor Mozetic, Miha Grcar, and Jasmina Smailovic. Multilingual twitter sentiment classification: The role of human annotators. *PLoS One*, 11(5):1–26, 2016.

[2] Ewan Klein Steven Bird and Edward Loper. *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009.

[3] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. On stopwords, filtering and data sparsity for sentiment analysis of Twitter. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, May 2014.

[4] Stanford NLP Group. *CoreNLP*. Stanford University, Stanford USA. https://stanfordnlp.github.io/CoreNLP/index.html.

[5] S. Vijayarani and R.Janani. Text mining: Open source tokenization tools – an analysis. *Advanced Computational Intelligence: An International Journal(ACII)*, 3(1):37–47, January 2016.

[6] M. Erwig and R. Gopinath. Explanations for regular expressions. In A. Zisman J. de Lara, editor, *Fundamental Approaches to Software Engineering*, volume 7212 of *Lecture Notes in Computer Science*, pages 394–408, Berlin, 2012. Springer.

[7] *Leveraging Inflection Tables for Stemming and Lemmatization*, volume 1, Berlin, January 2016. Association for Computational Linguistics.

[8] Amri Samir and Zenkouar Lahbib. Stemming and lemmatization for information retrieval systems in amazigh language. In M. Al Achhab N. Enneya Y. Tabii, M. Lazaar, editor, *Big Data, Cloud and Applications. BDCA 2018*, volume 872 of *Communications in Computer and Information Science*, pages 222–233, Kenitra, Morocco, August 2018. Springer, Cham.

[9] Anjali Ganesh Jivani. A comparative study of stemming algorithms. *International Journal of Comp. Tech. Appl*, 2(6):1930–1938, 2011.

[10] Anna Stavrianou, Caroline Brun, Tomi Silander, and Claude Roux. Nlp-based feature extraction for automated tweet classification. In *Proceedings of the 1st International Conference on Interactions between Data Mining and Natural Language Processing*, volume 1202, pages 145–146, September 2014.

[11] Ammar Ismael Kadhim. Term weighting for feature extraction on twitter: A comparison between bm25 and tf-idf. In *International Conference on Advanced Science and Engineering*, pages 1–6, Iraq, April 2019. IEEE.

[12] Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M. Ingersoll. An introduction to logistic regression analysis and reporting. *The Journal of Educational Research*, 96(1):3–14, 220.

[13] I. Rish. An empirical study of the naive bayes classifier. Technical report, T.J.Watson Research Center, Hawthorne, NY, 2001.

[14] Kaggle. Sentiment analysis on movie reviews. https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data, March 2014