**Natural Language Processing on SMS and Spam Filtering**

Matthew Afsahi

Informatics and Analytics, UNCG

IAF 603

 Dr. Lixin Fu

November, 14, 2020

**Abstract**

This paper discusses the basic of Natural language Processing, which is combination of machine learning technics with the text and also using statistics and math to format the data which how the machine learning algorithms can use. This project aims to predict the Spam texts and Spam messages with Spark big data engine using pyspark data frames technics.

**Dataset**

The SMS Spam Collection is a public set of SMS labeled messages that have been collected for mobile phone spam research.  A collection of 425 SMS spam messages was manually extracted from the Grumble text Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages.  A subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore. The messages largely originate from Singaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available.

**Attribute Information**

The collection is composed by just one text file, where each line has the correct class followed by the raw message. We offer some examples bellow:   
  
ham what you doing? How are you?  
ham Ok lar... Joking wif u oni...  
ham dun say so early hor... U c already then say...  
ham MY NO. IN LUTON 0125698789 RING ME IF UR AROUND! H\*  
ham Siva is in hostel aha:-.  
ham Cos i was out shopping wif darren jus now n i called him 2 ask wat present he wan lor. Then he started guessing who i was wif n he finally guessed darren lor.  
spam FreeMsg: Txt: CALL to No: 86888 & claim your reward of 3 hours talk time to use from your phone now! ubscribe6GBP/ mnth inc 3hrs 16 stop?txtStop  
spam Sunshine Quiz! Win a super Sony DVD recorder if you canname the capital of Australia? Text MQUIZ to 82277. B  
spam URGENT! Your Mobile No 07808726822 was awarded a L2,000 Bonus Caller Prize on 02/09/03! This is our 2nd attempt to contact YOU! Call 0871-872-9758 BOX95QU  
  
Note: the messages are not chronologically sorted.

**Features and Processing**

The main issue with the texts is that all the texts are in string format and there is no machine learning module that can analyze the strings. All the classification algorithms need some sort of numerical features vector in order to perform the classification tasks. There are many approaches to doing so, in this project I will be using Machine Learning Feature libraries to use Tokenizer, StopWordRemover, CountVectorizer, IDF, StringIndexer, VectorAssembler technics to tackle the raw.

[Tokenization](http://en.wikipedia.org/wiki/Lexical_analysis#Tokenization) is the process of taking text (such as a sentence) and breaking it into individual terms (usually words). A simple [Tokenizer](http://localhost:8888/notebooks/Downloads/DataScience_Jose_BigData/Spark_for_Machine_Learning/Natural_Language_Processing/api/scala/index.html#org.apache.spark.ml.feature.Tokenizer) class provides this functionality. [RegexTokenizer](http://localhost:8888/notebooks/Downloads/DataScience_Jose_BigData/Spark_for_Machine_Learning/Natural_Language_Processing/api/scala/index.html#org.apache.spark.ml.feature.RegexTokenizer) allows more advanced tokenization based on regular expression (regex) matching. By default, the parameter “pattern” (regex, default: "\\s+") is used as delimiters to split the input text. Alternatively, users can set parameter “gaps” to false indicating the regex “pattern” denotes “tokens” rather than splitting gaps, and find all matching occurrences as the tokenization result.

[Stop words](https://en.wikipedia.org/wiki/Stop_words) are words which should be excluded from the input, typically because the words appear frequently and don’t carry as much meaning. StopWordsRemover takes as input a sequence of strings (e.g. the output of a [Tokenizer](http://localhost:8888/notebooks/Downloads/DataScience_Jose_BigData/Spark_for_Machine_Learning/Natural_Language_Processing/ml-features.html#tokenizer)) and drops all the stop words from the input sequences. The list of stopwords is specified by the stop Words parameter. Default stop words for some languages are accessible by calling StopWordsRemover.loadDefaultStopWords(language), for which available options are “danish”, “dutch”, “english”, “finnish”, “french”, “german”, “hungarian”, “italian”, “norwegian”, “portuguese”, “russian”, “spanish”, “swedish” and “turkish”. A Boolean parameter caseSensitive indicates if the matches should be case sensitive (false by default).

[Term frequency-inverse document frequency (TF-IDF)](http://en.wikipedia.org/wiki/Tf%E2%80%93idf) is a feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus. Denote a term by $t$, a document by d, and the corpus by D. Term frequency $TF(t, d)$ is the number of times that term $t$ appears in document $d$, while document frequency $DF(t, D)$ is the number of documents that contains term $t$. If we only use term frequency to measure the importance, it is very easy to over-emphasize terms that appear very often but carry little information about the document, e.g. “a”, “the”, and “of”. If a term appears very often across the corpus, it means it doesn’t carry special information about a particular document. Inverse document frequency is a numerical measure of how much information a term provides:

𝐼𝐷𝐹(𝑡,𝐷)=log|𝐷|+1𝐷𝐹(𝑡,𝐷)+1IDF(t,D)=log⁡|D|+1DF(t,D)+1

Where |D| is the total number of documents in the corpus. Since logarithm is used, if a term appears in all documents, its IDF value becomes 0. Note that a smoothing term is applied to avoid dividing by zero for terms outside the corpus. The TF-IDF measure is simply the product of TF and IDF:

𝑇𝐹𝐼𝐷𝐹(𝑡,𝑑,𝐷)=𝑇𝐹(𝑡,𝑑)⋅𝐼𝐷𝐹(𝑡,𝐷).

CountVectorizer and CountVectorizerModel aim to help convert a collection of text documents to vectors of token counts. When an a-priori dictionary is not available, CountVectorizer can be used as an Estimator to extract the vocabulary, and generates a CountVectorizerModel. The model produces sparse representations for the documents over the vocabulary, which can then be passed to other algorithms like LDA. During the fitting process, CountVectorizer will select the top vocabSize words ordered by term frequency across the corpus. An optional parameter minDF also affects the fitting process by specifying the minimum number (or fraction if < 1.0) of documents a term must appear in to be included in the vocabulary. Another optional binary toggle parameter controls the output vector. If set to true all nonzero counts are set to 1. This is especially useful for discrete probabilistic models that model binary, rather than integer, counts.

**Models and Techniques**

It has been accepted that the Naïve Bayes approximation can be the most efficient method used for NLP (Edinburgh Informatics).This statically method estimates the p (document | class) which is the most proper approach for the text that has been researched. Therefore, the approach for analyzing this dataset is the Naïve Bayes which using the Spark’s ML Classification the Naïve Bayes gives the more accurate results for this project.

**Conclusion**

I initially started out with a raw dataset that contained two columns, one is the class column containing labels of Spam and Ham and also other column which is the raw texts corresponding to each class. After, applying many text feature extraction on the raw data as explained in features and processing section above, the data transformed to two numeric vectors columns that could be a proper numeric matrix to fit a Naïve Bayes model that is very suitable for text processing. Finally, as a result the model found to detect the texts and messages with the accuracy of the 92 % of Spam detection.

**References**

Almeida, T.A., GÃ³mez Hidalgo, J.M., Yama kami, A. Contributions to the Study of SMS Spam Filtering: New Collection and Results. Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11), Mountain View, CA, USA, 2011.

<http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/>

<https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>

<https://en.wikipedia.org/wiki/Feature_engineering>

<https://en.wikipedia.org/wiki/Bag-of-words_model>

<http://www.nltk.org/book/ch05.html>

<https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>

<https://en.wikipedia.org/wiki/Sparse_matrix>

<https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

<https://www.kaggle.com/c/word2vec-nlp-tutorial>