International Journal of Computing Academic Research (IJCAR)
ISSN 2305-9184, Volume 7, Number 4 (August 2018), pp.51-56
© MEACSE Publications
http://www.meacse.org/ijcar

# Traditional supervised weighting method vs. Logistic regression

Sajjad Salehi
DEIB – Department of Electronics, Information, and Bioengineering, Politecnico Di Milano Milan, Italy

## Abstract

Many text categorization algorithms have been developed during the past decade. These methods have been employed in many different aspects of life. In this work, we are comparing the accuracy of two well-known algorithms in text categorization. Which are supervised weighting method, and logistic regression machine learning algorithm. The corpus is related to a startup based in U.K., which is active in online e-commerce. Every offer in this corpus should be categorized either as a sale/purchase offer, hence the problem is a classical binary categorization.

**Keywords**: Machine Learning, Logistic Regression, Supervised Weighting, accuracy, text categorization, Multiclass classification, one vs. all algorithm.

## Introduction

There are many different developed algorithms to be used as text categorizer. Some conventional ones include not only weighting method and logistic regression but also some other simpler or more complex algorithms such as n-gram analysis, a bag of words analysis and a-priory. Many types of research have shown strength and weakness of each method. In this paper, we chose two methods which have the least complexity and used more often in commercial areas to compare their accuracy, as well as challenging their robustness for the commercial use.

Man LAN and Chew Lim [1] have discussed many different methods for automatic text categorization generally based on text weighting. Text weighting methods are one of the most robust text categorization methods. They have been around for many years and their simplicity to understand made them the first choice to employ for many start-ups.

Some different factors employed in text categorization [2]:

1. Term frequency Factor

Commonly used frequency factors include parameters in a vector which allows mapping a text to a vector space. They usually include a binary factor which represents the presence of a word, term frequency alone which shows the how many times a word occurs in a document Factors include:

- Term frequency factor: binary
- Term frequency (tf): number
- The logarithm of term frequency: log (1+tf)
- Inverse term frequency (ITF): usually r = 1 (formula 1)

$$1-\frac{r}{r+tf}\,(1)$$

2. Collection Frequency Factor

- TF: term frequency
- IDF: Multiply TF by an inverse document frequency (IDF) factor
- IDF probability: Multiply TF by a term relevance, i.e. probabilistic IDF
- Chi square: Multiply TF by a chi-square function
- Gain ratio: Multiply TF by a gain ratio function
- Odds ratio: Multiply TF by an Odds Ratio function
- Normalization Factor

The term frequency factor is enough to use as term weights without other factors. In another study [3], three term factors are studied to investigate if term frequency alone methods i.e. tf, log (1+tf), ITF could work in linear SVM in terms of micro advantage. The results show that there is no significant difference among them. The significant reason is that all these parameters are derived from the same idea which is the term occurrence. As a result, we decided to use only TF alone as an input parameter for the Machine Learning method. Moreover, many more different variables are used in many different studies. In [4] logarithmic operations are studied to scale down the unaffordability of the high term frequencies in one document which then led to inverse term frequency introduced by [5].

Since term frequency alone may not have the discriminating power to pick up all relevant documents from other irrelevant documents, an IDF (inverse document frequency) factor which takes the collection distribution into account has been proposed to help to improve the performance of IR in [6]. The IDF factor varies inversely with the number of documents ni which contains the term ti in a collection of N documents and is typically computed as log (N/ni).

In a nice similar research [7], Ifrim, Bakir, and weikum have shown that the logistic regression has a good impact in categorizing documents using variable length n-gram words or characters while learning involves automatic tokenization. They tried to solve this problem using n-gram logistic regression using gradient ascent approach. At the end, they proposed that branch and bound approach, which chooses the maximum gradient ascent.

## The Problem

The problem is almost well defined. There is a data set of almost 267 thousand purchases online, on the e-commerce platform. All the provided data is in plain text, and there is no pre-defined tag.

Since the owners wanted to keep it as simple as possible for the users, no extra information is asked from the users, unless they are willing to add such information in which unfortunately in most cases users have not added enough information about their offers on the platform. In the first step to solving many problems for this startup, we decided to categorize the offers into purchase/sale offers, which could be a starting point and a key to solving many more problems. Our future works are related to the same data set, trying to solve different problems with different aspects.

## Data Structure

The Dataset includes data about objects as used by the home company. Here there is an overview of what kind of objects there is in the data set:

*A. Users*

Users are simply the ones who use the platform. Since the platform is designed for the students, we assume almost all the users are students. This object has properties like name, gender, university, location, picture, etc.

*B. Offer*

The offer is simply a text, which indicates the product, or service that the offer poster wants to buy or sell. This object has properties like date, message, pictures, user, price, location, likes, comments, etc.

## General Statistics about the Corpus

It is nice to know some statistics about the data set we are working on. Please consider that the data set provided was noisy, there was a real need to clean the data before working on it. During the labeling period, we did our best to have a dataset as clean as possible.

A. *Users on the platform*

There are almost 263 thousand users on the dataset provided. In which more than 56% of them are male users (Fig. 1) and about 84% of them are located inside Italy (Fig. 2).

B. *Offers on the data set*

There are many offers in the dataset. In total, they are around 267 thousand offers and the people who are interested in that offer as well as their comments. The offer includes a text field which all the information should be extracted from. Everyone writes whatever he wants without any restriction. This offer could also contain pictures, which are a good source to extract information.

C. *Pictures*

There are plenty of pictures in the dataset. At the moment we decided not to use the pictures directly on decision making, but using them as a parameter of our categorization. We can use the number of pictures –which also can be zero-as an input besides other inputs we have for our learning algorithm.

D. *Price*

Price is also another parameter that we have used as an input parameter for our Machine Learning system. Users are free to enter the price or not. We tried to see if the price can guide us to categorize the text or not.

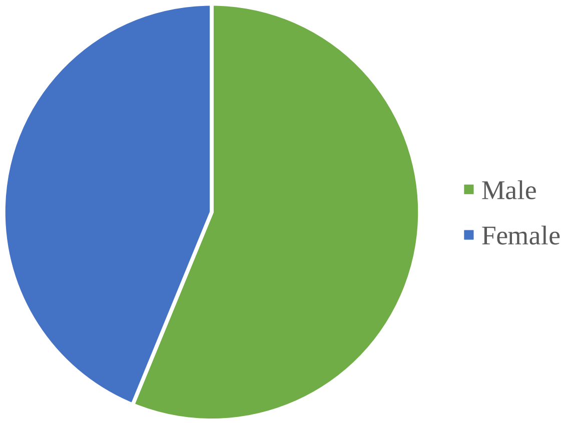

*Figure 1Gender Distribution over the platform*

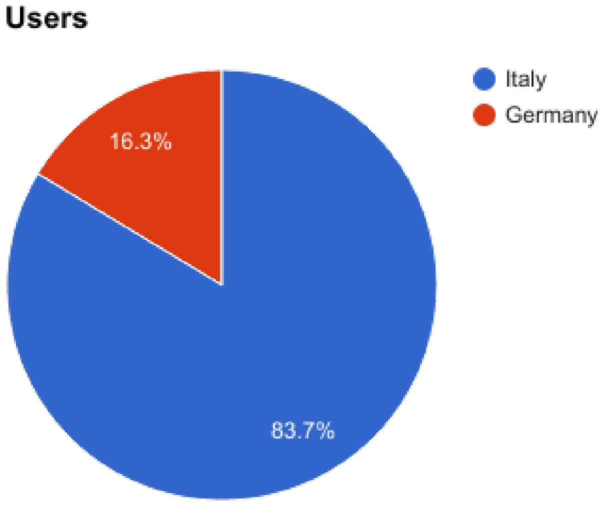

*Figure 2Distribution of users over countries*

## Problem-Solving Approach

The standard method of word tokenization is commonly used in text categorization as a means of the training set before the learning algorithm. Obviously, there should be also some language dependent pre-processing as well. This includes removing the stop words and stemming [8]. This part is crucial for our research since our data set contains different languages including “Italian”, “German”, and “English”. It is often crucial to do such a token engineering, which also needs an expert of the language to be categorized. Furthermore, this is highly useful in tuning the classifiers with unigram features. In contrast, there are some important classification tasks in which the initial unigram bag of words does not help a lot in solving the problem even if the learning algorithm itself is very powerful [9, 10, 11, 12, 13].

Examples are: email categorization, sentiment analysis mining in product reviews, user classification in social networks, and classification in social communities. For these kinds of applications, some more complex features like word n-grams or natural-language parse tree are needed in order to fine tune even more the results.

However in a research by Kadu, and Taku [15] It is approved that performance and quality of the results of n-grams do not have too much difference with a highly complex NLP parse tree which is always more complex than an n-gram machine learning based algorithm. Hence we decided to have a machine learning n-gram approach to solve this problem while the quality of the results is not that different from the ones from an NLP parse tree.

*A. First phase*

As the first phase, we decided to tag around 1000 offers for the machine learning algorithm input. This tagging phase includes only a purchase/sale tag. In this phase, we tried to be as general as possible. The offers are considered as three different values, purchase requests; sell requests, and unknown requests. There are some requests in the dataset, which we decide to label as unknown because the intention of the user is not clear whether he wants to sell a product, or he wants to buy something.

At the end of 1000 labeling, there were 23.3% purchase requests, and 76.7% are sell requests.

*B. Second phase*

Pricing the offers. As the data-set is text only, there were no precise data about the prices price included. Therefore we decided to introduce a very simple greedy algorithm to find the price of each offer. The algorithm is pretty simple. It looks for any number (text or numeric character) and considers it as price. Since there are almost no more numbers in the offer texts, it worked pretty fine. At the end of the second phase, we entered price data as labels and compared it with the previous data coming from the greedy algorithm. As we finished the labeling of the first 1000 offers, we found out that there was only error in 16 rows out of 1000, which gets the accuracy of 98.4%. However, it is not a great accuracy to consider, it is not bad as well. Therefore, we decided to keep the same algorithm for the whole project.

*C. Third phase*

In this phase, we started to use the supervised weighting algorithm. We tried to extract the words, whichare used by the users in the offers and tag the ones, which have the weight for the purchase or sell tagging. Obviously, the words are in different languages, mostly in Italian, German, and English. You can see the most used words we found and their frequency in the whole corpus in table 1.

*Table 1sample of n-gram weights*

| Word | Frequency | n-gram |
|---:|---:|---:|
| **Vendo** | 83380 | Unigram |
| **Suche** | 59317 | Unigram |
| **Cerco** | 36593 | Unigram |
| **Suchen** | 10845 | Unigram |
| **Looking for** | 10214 | Bigram |
| **Sell** | 2460 | Unigram |

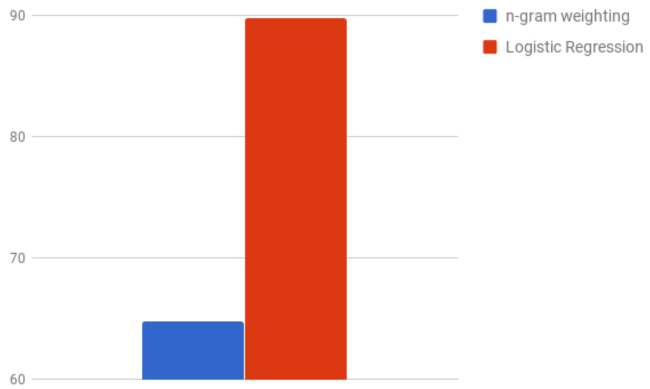

*Figure 3Accuracies*

As a normal term weighting algorithm, we chose the formula 2. It's simple to understand, yet powerful enough to get you the results.

$$\sum_{i=1}^{n}[(-w_i)(cat_1) + (w_i)(cat_2)] / n \quad (2)$$

In this Formula, there are a few parameters.

- w is the word or n-gram in the text.
- i: the i<sup>th</sup> n-gram we are looking at, in the text.
- n: number of the n-grams in the text.
- cat is a binary parameter with the values of {0, 1} which indicates if the word belongs to.

This formula returns a number between [-1, 1] which indicates the category that the text belongs to. Reaching -1 means that the algorithm is almost certain that the text falls into category 1, and 1 means the opposite.

Based on the number of the n-grams we had in the dataset, we decided to use only unigrams, bigrams, and trigrams. By eliminating other n-grams, we could save a lot of time, and computing power. Our whole dictionary including unigram, bigram, and trigrams consists of around 4.5 million records, which is hard to process.
At the end of this phase, we decided to calculate the accuracy of this method. Based on each correct guess by the algorithm, it receives one point. The total points this algorithm could achieve based on the prepared words bag was 648. Which means the accuracy of 64.8%, which is a fair accuracy to consider. Keep in mind that we used the very basic term weighting algorithm that is fast in runtime and easy to develop.

*D. Fourth phase*

In this phase, we started to categories the sentences using the Logistic regression in order to categorize the offers we have in the dataset. For this purpose, we used the labels we provided in the first phase. As a simple logistic regression problem, we used the simple formula 3.

$$h_\theta(x) = g(\theta^T x) \qquad (3)$$

In which the g function is the Formula 4.

$$g(x) = \frac{1}{1 + e^{-\theta^T x}} \qquad (4)$$

In formula 3, the hypothesis h shows the probability of the text to be belonging to the category 1. For example h(x) = 0.7 shows that the probability to be in category 1 in 0.7, and being in category 2 is the complent, which is 0.3. The g function is a regular sigmoid function which enables us to distinguish a function which bounds between 0 and 1. To put it in simple words, category 0, and 1. In all these formulas theta is the guess matrix which we are going to have in our hypothesis. The T super script is the transpose sign. The e in the g function is the Euler number.
As the cost function, the function 5 is introduced. This formula penalizes infinite in case the category is not chosen with 100% certainty, and brings down the penalty as the wrong guess has lower certainty.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] \qquad (5)$$

The formula 5 contains parameter "m" which is the number of samples.
In here about the training set, we provided a bag of n-grams contained in the offer using a binary flag which indicates the n-gram's presence in the offer as well as its frequency in the offer and also in the whole training set.

At the end of this process, we started to calculate the accuracy and compare it with the previous algorithm provided. The pointing system is exactly what defined for the previous algorithm. At the end, this algorithm could guess 897 offers correctly which gives us the accuracy of 89.7%. Considering that this algorithm is also a very simple algorithm in its kind with no fine tune, it’s a good result for a classifier.

## Conclusions

In this paper, we have shown that however, the supervised n-gram weighting algorithms have a fairly good accuracy, but a simple Logistic Regression method can boost the accuracy by almost 25%. You can also consider if you fine-tune it, the accuracy will be even more. (Figure 3)

Furthermore, the accuracy around 65%, which is gained by n-gram weighing, is not acceptable in the real business world. However the 89.7% gained by Logistic regression is not the best result ever, it’s quite acceptable on the market.

## References

[1] Man Lan, Chew Lim Tan, “Supervised and Traditional Term Weighting Methods for Automatic Text Categorization”. JOURNAL OF IEEE PAMI, VOL. 10, NO. 10, pp. 721–735, July 2009.

[2] Salton, Gerard, and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." Information processing & management 24.5 (1988): 513-523.

[3] Lan, Man, et al. "A comparative study on term weighting schemes for text categorization." Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on. Vol. 1. IEEE, 2005.

[4]Buckley, Chris, et al. "Automatic query expansion using SMART: TREC 3." NIST special publication sp (1995): 69-69.

[5]Leopold, Edda, and JörgKindermann. "Text categorization with support vector machines. How to represent texts in input space?." Machine Learning 46.1-3 (2002): 423-444.

[6]Wu, Harry, and Gerard Salton. "A comparison of search term weighting: term relevance vs. inverse document frequency." ACM SIGIR Forum. Vol. 16. No. 1. ACM, 1981.

[7]Ifrim, Georgiana, Gökhan Bakir, and Gerhard Weikum. "Fast logistic regression for text categorization with variable-length n-grams." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008.

[8] Holmes, David I., and Richard S. Forsyth. "The Federalist revisited: New directions in authorship attribution." Literary and Linguistic computing 10.2 (1995): 111-127.

[9] Kessler, Brett, Geoffrey Numberg, and HinrichSchütze. "Automatic detection of text genre." Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 1997.

[10] Lee, Yong-Bae, and Sung HyonMyaeng. "Text genre classification with genre-revealing and subject-revealing features." Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2002.

[11] Peng, Fuchun, Dale Schuurmans, and Shaojun Wang. "Augmenting naive bayes classifiers with statistical language models." Information Retrieval 7.3 (2004): 317-345.

[12] Zhang, Dell, and Wee Sun Lee. "Extracting key-substring-group features for text classification." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006.

[13] Yang, Yiming, and Jan O. Pedersen. "A comparative study on feature selection in text categorization." Icml. Vol. 97. 1997.

[14] McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." AAAI-98 workshop on learning for text categorization. Vol. 752. 1998.

[15] Kudo, Taku, and Yuji Matsumoto. "A Boosting Algorithm for Classification of Semi-Structured Text." EMNLP. Vol. 4. 20