International Journal of Engineering & Technology, 7 (4.35) (2018) 234-238

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



Term Weighting Vs. Logistic Regression Performance on E-Commerce Data

Sajjad Salehi[1]*, Maryam Ghasdimanghootai[2]

[1]Politecnico di Milano, Italy
[2]Parand Islamic Azad University, Iran
**Corresponding author E-mail: sajjad.salehi@mail.polimi.it*

Abstract

Text categorization can become a very difficult problem to solve in many cases. However many text categorization algorithms have been developed in the history of computer science, they are not always as accurate as we expect. Some of them are highly accurate in special cases while others perform well in different cases. In this work, we are comparing two famous methods in text categorization; the first one is the well-known term weighting algorithm and the second one is the logistic regression algorithm. All the dataset is got from our previous start-up named "Ume Market Network" which was an online peer-to-peer e-commerce system, and was synchronized with Facebook sales groups. Every offer in this dataset should be categorized as a sale/purchase offer; therefore, the problem is a classical binary categorization on a text dataset of formal as well as colloquial expressions in English, Italian, and German languages. After overcoming all the ambiguities the logistic regression algorithm outperformed the term weighting algorithm by around 25% in acuracy.

Keywords: *Machine Learning; Logistic Regression; Supervised Term Weighting; Text Categorization; Multiclass Classification*.

1. Introduction

There are many different developed algorithms to be used as text categorizer. Some conventional ones include not only weighting method [1, 2] and logistic regression [3, 4] 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 [5] 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 [6]:

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)

$$1 - \frac{r}{r + tf} \tag{1}$$

2. Collection Frequency Factor

In collection frequency, there are much more detailed parameters to consider. As an example original term frequency, IDF (inverse document frequency) and IDF probability (term relevance) are considered in this method. The factors used are:

- 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 [7], 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 [8] 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 [9].

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

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

to improve the performance of IR in [10]. 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 [11], 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.

3. The Problem

The problem is almost well defined. There is a dataset 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 start-up, 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 will be related to the same dataset, trying to solve different problems with different aspects.

3.1. 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:

3.1.1. 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.

3.1.2. Offers

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.

4. General Statistics about the Dataset

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 labelling period, we did our best to have a dataset as clean as possible.

4.1. 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).

4.2. Offers on the Platform

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.

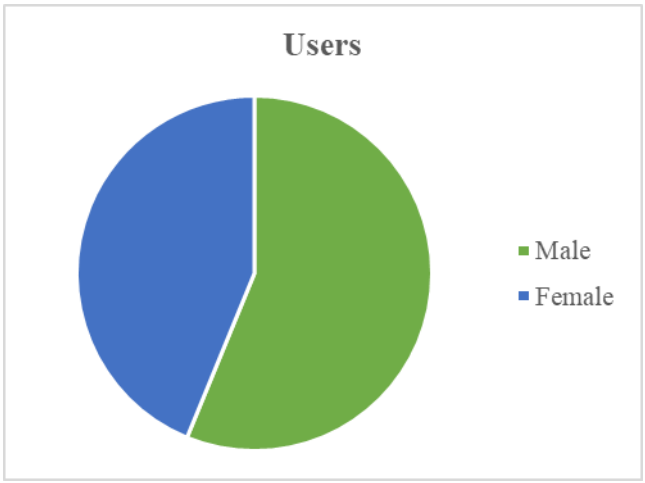


Figure 1: Gender distribution over the platform

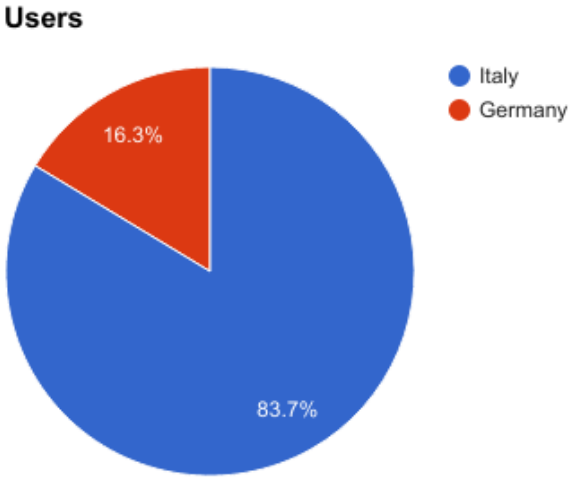


Figure 2: Distribution of users over countries

4.3. 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.

4.4. 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 could guide us to categorize the text or not.

5. 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 [12]. 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 classi-

fiers 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 [13, 14, 15, 16, 17].
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 [18] 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 dataset 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 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 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, which are 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 dataset in table 1.
As a normal term weighting algorithm, we chose the formula 2. It is 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^{th} 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.

Table 1: sample of n-gram weights

Word	Frequency	n-gram
Vendo	83380	Unigram
Suche	59317	Unigram
Cerco	36593	Unigram
Suchen	10845	Unigram
Looking for	10214	Bigram

Based on the number of the n-grams available in the dataset, we decided to use only unigrams, bigrams, and trigrams. By eliminating other n-grams a lot of time and computing power could be saved. The whole dictionary including unigram, bigram, and trigrams consists of around 4.8 million records, which is hard to process.
We performed a very simple algorithm using greedy term weighting formula. This formula uses a finite dictionary, which shows predefined N-grams tags and weights the text using Formula 2. In that formula, there are only two categories (here I use them for Buy/Sell tags) and the dictionary is populated with the data in Table 5.
Each word that is found in the dictionary will be kept in the formula, and the ones, which are not in the dictionary, will not be applied in the formula. The category1 is dedicated to the buy offer, and the category2 is dedicated to the sell offers.
This algorithm really surprised me when it performed well on categorizing 200 test data. It could determine 158 correct categorizations versus 42 incorrect categorizations. Therefore the overall accuracy of this algorithm turned out to be 79%.
However the accuracy of 79% is almost fine, but I decided to go further for a more stable algorithm which can perform always well in any condition. Therefore, I went for a modified version of the previous algorithm. With a supervised term weighting algorithm that performs better than the first version. The 1000 training set are used as the dictionary for the input of this algorithm. These data include these fields:

- Frequency is the frequency of this N-gram used in the tagged corpus.
- Buy Frequency: the frequency of this N-gram used as a buy offer.
- Sell Frequency: the frequency of this N-gram used as a sell offer.
- Sell Percentage: usage of this N-gram as a sell offer in the whole tagged corpus.

We used this supervised dictionary as an input for the same formula and tested the 200 test data to determine the accuracy. Unfortunately, this improvement did not work well. On the test data, the accuracy of this algorithm turned out to be on 77.5%, which is slightly lower than the result I obtained on the simple version of the algorithm.
The best accuracy this algorithm could achieve was 79%, which is a fair accuracy to consider. Keep in mind that we used the very basic term weighting algorithm that is fast at runtime and easy to develop.

D. Fourth Phase

In this next step, we went for a more reliable algorithm, which could be a more complex algorithm. This algorithm is a logistic regression using a sigmoid function to categorize the data in only two categories (Buy/Sell). For this purpose, we used the labels provided in the first phase. As a simple logistic regression problem, I used the simple Formula 3.

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

In which the g function is the Formula 4.

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

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, or 1 will be chosen using this formula. In all these formulas theta is the guess matrix which we are going to have in our hypothesis. The T superscript is the transpose sign. The e in the g function is the Euler number.
As the cost function, the Formula 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.

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

The Formula 5 contains parameter "m" which is the number of samples.
As the training set, a bag of n-grams is provided 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.
The parameters of the Machine Learning system is the following:

- Input:
  - Price of the offer
  - Number of the pictures
  - N-grams which are used in the offer
- Output
  - Buy/Sell category

We decided to use the n-gram used in the offer in order to direct the Machine Learning algorithm in a way to find relations between the words used in the offer and the output. The algorithm is developed using Matlab.
The gradient descends for such a thing would be Formula 6.

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \quad (6)$$

It is always good to visualize the data if it is possible; in this case, with high dimensions of the data, it is not easy to visualize the data,
However, we have tried to show the portion of the data, which is more interesting. Figure 3 shows the input training data for the Logistic Regression set which are labeled in Buy/Sell. On the axis, you can see the offers are distributed based on the number of pictures and the price. Every sell offer is shown with black crosses, and every buy offer is shown in yellow circles. In this visualization, it is obvious that almost all the buy requests are without pictures, but they are in any price range.

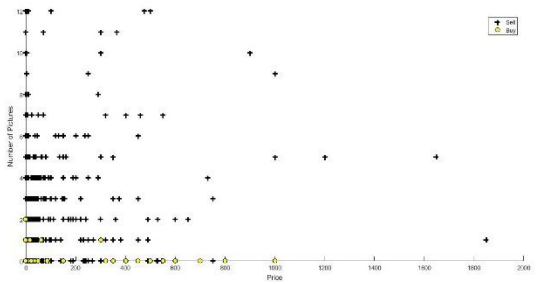


Figure 3: Number of pictures vs. the price from an offer

At the end of the learning phase, the algorithm determined the decision boundary, and again we tried to show it only in two dimensions in Figure 4. This decision boundary shows if you consider only the price and number of pictures, the offers which fall below the decision boundary line are buy requests, while the offers which fall on the upper side of the line, are the sell offers. This decision boundary shows that in this 2-dimensional space the machine-learning algorithm has decided to identify the offers without pictures as buy requests, and eliminates the other buy requests, which have some pictures included because of the high density of sell requests with pictures.

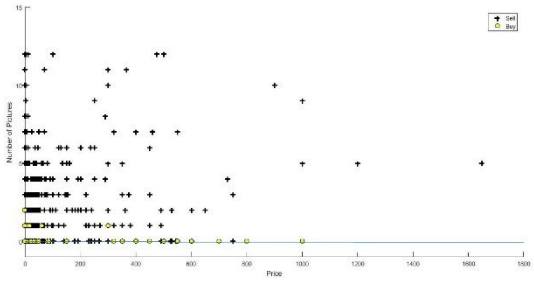


Figure 4: decision boundary on training set

Figure 5 shows the optimality of the Matlab's fminunc function during the iterations; it is demonstrated on the picture that the function has finished the work after 13 iterations, but it has almost converged after the eighth iteration.

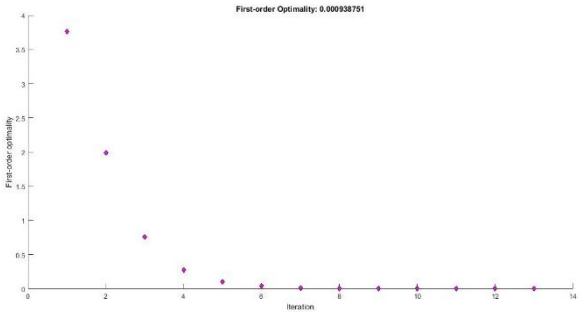


Figure 5: optimality in the iterations

At the end of this process, the accuracy of the algorithm is evaluated as 89.5%, which is a good enough accuracy to predict the next offers.
Figure 6 shows the comparison of the accuracies over the categorization methods used in this paper.

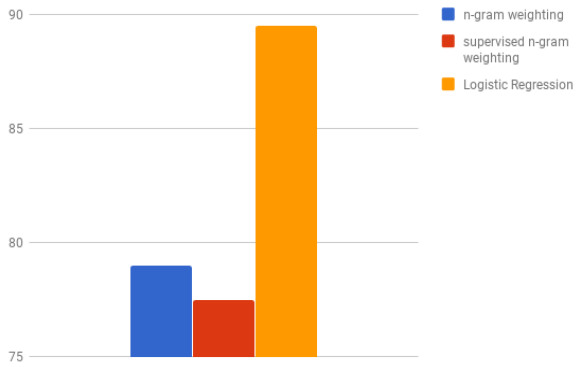


Figure 6: Accuracy of the classifiers

6. Conclusion

In this paper, a very basic term-weighting algorithm and a modified version of n-gram term weighting algorithm are compared with a very simple logistic regression. The algorithms are applied to our peer-to-peer e-commerce network.

At the end, we have shown that however, the supervised n-gram weighting algorithms have good accuracies, but a simple Logistic Regression method can boost the accuracy by almost 25%. You can also consider a fine-tuned algorithm can achieve even a higher accuracy. (Figure 6).
Furthermore, the accuracy around 79%, which is gained by n-gram weighing, is not acceptable in the real business world. However the 89.5% gained by Logistic regression is not the best result ever, it is quite acceptable on the market.

References

[1] El-Khair I.A. Term Weighting. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA, pp.131, 2009.
[2] Hofmann, Thomas. "Probabilistic latent semantic indexing." ACM SIGIR Forum. Vol. 51. No. 2. ACM, 2017.
[3] Harrell, Frank E. "Ordinal logistic regression." Regression modeling strategies. Springer pp. 311-325, 2015.
[4] Cruyff, M. J. L. F. "A review of regression procedures for randomized response data, including univariate and multivariate logistic regression, the proportional odds model and item response model, and self-protective responses." Handbook of Statistics. Vol. 34. Elsevier, pp. 287-315. 2016.
[5] 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.
[6] Salton, Gerard, and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." Information processing & management 24.5, pp.513-523. 1988.
[7] 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.
[8] Buckley, Chris, et al. "Automatic query expansion using SMART: TREC 3." NIST special publication sp (1995): 69- 69.
[9] Leopold, Edda, and Jörg Kindermann. "Text categorization with support vector machines. How to represent texts in input space?." Machine Learning 46.1-3 (2002): 423-444.
[10] 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.
[11] 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.
[12] Holmes, David I., and Richard S. Forsyth. "The Federalist revisited: New directions in authorship attribution." Literary and Linguistic computing 10.2 (1995): 111-127.
[13] Kessler, Brett, Geoffrey Numberg, and Hinrich Schütze. "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.
[14] Lee, Yong-Bae, and Sung Hyon Myaeng. "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.
[15] Peng, Fuchun, Dale Schuurmans, and Shaojun Wang. "Augmenting naive bayes classifiers with statistical language models." Information Retrieval 7.3 (2004): 317-345.
[16] 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.
[17] Yang, Yiming, and Jan O. Pedersen. "A comparative study on feature selection in text categorization." Icml. Vol. 97. 1997.
[18] 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.
[19] Kudo, Taku, and Yuji Matsumoto. "A Boosting Algorithm for Classification of Semi-Structured Text." EMNLP. Vol. 4. 2004