Comparison of tokenizers on text mining using text-based method in gender prediction

SCC-413 Apply Data Mining
Wang Xinji 32026312

Abstract

The research produce a guideline in text mining area of how text data is transform to numerical data. It use text-based method and compare several traditional tokenizers, along with three machine learning algorithms to reach a conclusion that a shorter but meaningful corpus could decrease the runtime of model training while not losing much accuracy.

1. Introduction

One principle job of social media website and other product sales website is estimating user behaviors and user information base on the data collected from social media like twitter, facebook and other media social products. Part of this topic is estimating the users' age, gender and their interest hence suitable products or contents are recommended to bring in financial income. But the information collected from social media is not usually present as numerical data but text and pictures. Thus, we propose a framework to initially positioning the users of the website, start from predicting the age of a user base on the tweeter text a user post in the social media.

Various related work has been released to portrait a user via text mining [1] or produce sentiment analysis [2]. In this paper, instead of complex sentiment analysis or user interest prediction, we focus on the estimating the basic information of a user using various of machine learning algorithm and tokenizer combined with a self-produced vocabulary word embedding (a default vectorizer will also be used as a comparison) on a British celebrities dataset, where the dataset include some basic information of the user and ones' text of twitter content as well [1].

2. Related work

Automatically summarizing information from the website has been attempted for a long time, and most of people tend to use text-based classification [1].Although the target estimated of each scholar differs from gender, age[3], area[4], education level[5] and personality.

Besides, the further study moves from basic information estimation to a more detailed area, trying to unearth customers' behavior and favor, as well as ones' sentiment and opinion [6] or influence to the community.

All the above-related works follow a standard procedure, start from transforming text content into numerical vectors and then implement machine learning algorithm, in spite of some differences in detail (procedure when cleaning data, usage of vectorizer tool and models used).

When it comes to modern framework, correlations between context (connection of context) and the order words appear is considered. Usually by introducing LSTM,RNN model and Markov chain ([8],[9]).

3. Methodology

3.1 Introduction of datasets

The dataset used is a large collection of verified "celebrities" included 79 users on twitter, this dataset can also fetch in the Wikipedia. Among all the users, 26 female users and 49 male are identified, where 4 unknown values presented.

[1]dataset source :https://modules.lancaster.ac.uk/pluginfile.php/1598262/mod_page/content/13/celebs.zip
[2] explanation quote from :http://www.nltk.org/api/nltk.tokenize.html

The dataset also provides the information of language, age, age range, user-id and the time to send each tweet (source of dataset see footnote[1]). And despite the dataset provide so many basic information, this research focuses on only digging information from the text hence the only variable entering the model are all comes from the text. Due to the number of observations, the result after cross-validation will be affected; this problem will be discussed in section 3.2.

3.2 Text-based method

The design of the pipeline follows a traditional procedure as we discussed in section 2. Start from step-1: collecting data then we clean the data, mostly focusing on cleaning the data, clean out the 'HTTP' link quoted in the tweets (both web links and images link), emoji used and the symbols and also we lower all the characters in the text to avoid double count of words. All the data is prepared and stored in the format of data frame (step-2), along with frequency analysis of the texts (step-3). After that comes to step-3, we process segmentation and use different tokenizers to create our bag-of-words (from the data source), and exam several tokenizers on it. Finally, we choose a LinearSVC, logistic regression classifier and a random forest classifier to compare the different bag of words, tokenizers to reach our result. And also, we produce cross validation to split our data set into training data and testing dataset to exam how well our model performs. This final step can regard as a kind of text classification characterizing a user.

On evaluating the research result, we use confusion matrix and produce sets of criteria in machine learning to compare the models, which are accuracy, precision, recall and f1-score. Due to the size of dataset, the size of test set is also limited, but to avoid over-fitting (which we will see in the result part), 80% of data is used to train and only 20% of data is used to exam our model (around 14), this will cause huge fluctuation of our exam result inevitably.

3.3 Introducing tokenizer

The first tokenizer introduce is a simple white space tokenizer, the characters in the text will be automatically split from space to space; this is one of the simplest tokenizers hence it will be set as the baseline of the model.

After the white space tokenizer, we introduce a tweeter tokenizer from the natural language toolkit (nltk). This is defining as: *'Twitter-aware tokenizer, designed to be flexible and easy to adapt to new domains and tasks* [2]*'*. This tokenizer can detect flexible word including symbolic facial expression; the Built-in options in this tokenizer allow us to get the lower case of text [2].

The last tokenizer we introduce is the word-punctuation tokenizer, which allows we split text according to punctuation in tweets. This tokenizer will join as a competitor in bring in more uncertainty.

3.4 tf-idf

Tf-idf or, term frequency-inverse document frequency is a traditional method transforming text data into numerical data set so we can process numerical analysis on a text data.

The first part is called term frequency, which evaluates the time that words appear in the document, but these could usually be meaningless, probably words like: "the", "is", and so forth. To avoid this draw back, the inverse document frequency is used. It decreases the weight for commonly used words above and increases the importance of a word that appears in a particular document that not appears in the other document. Combing two things together forms the tf-idf algorithm. Under our research, after tokenize the texts; we use this method to create our numerical vector.

Having $n_{i.j}$ is the frequency of words in document d_j, |D| is the number of documents, and j is number of documents that contain one particular words, the equation of tf-idf is as follows:

[1]dataset source :https://modules.lancaster.ac.uk/pluginfile.php/1598262/mod_page/content/13/celebs.zip
[2] explanation quote from :http://www.nltk.org/api/nltk.tokenize.html

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}, \qquad \mathrm{idf_i} = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

And: $$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i} \qquad (1)$$

4. Results

4.1 frequency analysis

The frequency analysis gives a preliminary inspect on how each tokenizer differs. The common point from the frequency plot of the full-text data shows that the frequency follows the zip's distribution, where the frequency of the top 10 words dominant the whole text data (he results from other tokenizer shows the same result so is not shown here).

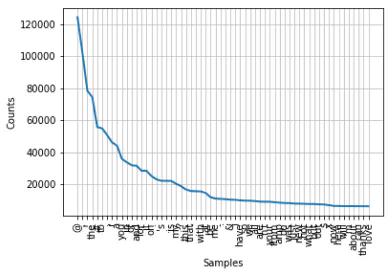
Figure 1 frequency plot

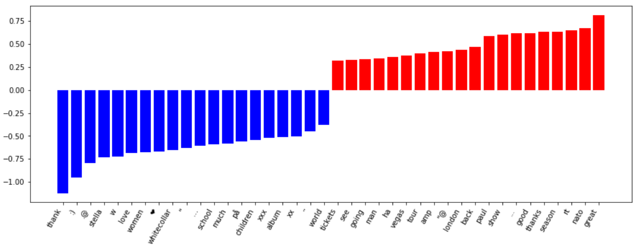
Figure 2 words coefficient plot

Table 1 tokenizer result

	White-space	nltk.word_tokenize(default)	Twitter	word-punctuation	self_produce tokenizer
average token length	4.615	3.993	4.256	3.91	9.2169
total number of token	2513253	2901860	2708405	2961109	99277

Despite the consistency of the frequency plot, a small difference is detected in the number of tokens found from the text and hence the average length of tokens differs. In table-1, we can see the word-punctuation tokenizer extract the largest number of words, and the white space extracts least words in the data. But the number of words it takes does not represent the performance of a tokenizer. Especially we produce a word vocabulary from the dataset and make some cleaning and remove those meaningless words, reducing the size of vocabulary can raise the runtime while training data and may be robust when facing noisy data. Another reason bringing in this tokenizer is when considering migrate the phrases database; it can be used in different project and datasets. So this tokenizer acts as an attempt in the project.

4.2 model result

From figure-2, as label '1' stands for male, we can see males tend to use words such as 'great', 'good' and talk about 'man', while females uses symbol facial expression, 'love' and love pattern and also talk about 'women'. But both group shares some common words like 'thanks' and so on.

From table we could assert that model set-1, over fitted does exist while we train and test using full datasets. All the performance dominant all other results and especially random forest-1 got a 100% correct rate. But the drawback of the default tokenizer is that it took the longest time, while the self-produced tokenizer used only 48s. Among the three models, it is hard to find out the best model, all the performance criteria fluctuated during every training loop. So after 10-cross validation, we get the mean accuracy and other corresponding metrics.

[1]dataset source :https://modules.lancaster.ac.uk/pluginfile.php/1598262/mod_page/content/13/celebs.zip
[2] explanation quote from :http://www.nltk.org/api/nltk.tokenize.html

Table 2 Model result

tokenizer	model	precision	recall	f1-score	accuracy	runtime
nltk.word_tokenize(default)	Linear SVC-1*	0.96	0.96	0.96	0.96	112s
	Logistic classifier-1*	0.84	0.79	0.76	0.79	
	random fores-1*	**1**	**1**	**1**	**1**	
White-space	Linear SVC-2	0.62	**0.79**	0.69	0.78	50s
	Logistic classifier-2	0.62	**0.79**	0.69	**0.785**	
	random fores-2	**0.65**	0.57	0.6	0.57	
nltk.word_tokenize(default)	Linear SVC-3	**0.83**	0.79	**0.8**	0.78	114 s
	Logistic classifier-3	0.73	**0.86**	0.79	**0.85**	
	random fores-3	0.81	0.71	0.75	0.71	
Twitter tokenizer	Linear SVC-4	0.81	0.79	0.8	0.78	102s
	Logistic classifier-4	**0.88**	**0.86**	**0.83**	**0.85**	
	random fores-4	0.81	0.79	0.8	0.78	
word-punctuation	Linear SVC-5	**0.78**	**0.79**	**0.78**	**0.93**	49s
	Logistic classifier-5	0.41	0.64	0.5	0.85	
	random fores-5	0.71	0.64	0.56	0.71	
self_produce tokenizer	Linear SVC-6	0.62	0.79	0.69	0.78	48s
	Logistic classifier-6	0.62	0.79	0.69	0.78	
	random fores-6	**0.84**	**0.79**	**0.74**	**0.78**	

*Model set 1 is train and test without cross validation

From table-2, the self-produced tokenizer which uses the word embedding directly from the text outperforms the white-space tokenizer, and it basically has a close performance compare to the other three tokenizers. And one advantage is that the words list creates from the text is cleaned, which means it can be preserved and used as corpus and work in a new data set. Considering the twitter tokenizer, it is just slightly better than others, but cost almost the longest time in training although it is designed to work on twitter texts.

5. Discussions

In this research we examine three tokenizers in the toolkit and create a self-produced word embedding, use these word corpus, combined it with tf-idf algorithm, we reach a satisfiable result with 70%+ accuracy and 80%+ of the metrics in gender prediction. Simultaneously, we prove that a shorted word corpus could actually reduce the runtime of training a model while it brings in convenience of preserved as a word corpus in a certain area, which matches our expectation before the experiment as the frequency of matching a word in corpus reduces.

However, more tokenizer is still available to used, and the other information in the dataset is not used, consider the text information from usernames and frequency of emoji usage, the performance of our model is yet to raised. Furthermore, limited by the size of data, the result fluctuated severely, if more data is collected using web crawler, the research result will be more convincible. These insufficient can be listed as part of future works.

6. References

[1]Ikeda K, Hattori G, Ono C, et al. Twitter user profiling based on text and community mining for market analysis[J]. Knowledge-Based Systems, 2013, 51: 35-47.

[2]Wang X, Wei F, Liu X, et al. Topic sentiment analysis in twitter: a graph-based hashtag

[1]dataset source :https://modules.lancaster.ac.uk/pluginfile.php/1598262/mod_page/content/13/celebs.zip
[2] explanation quote from :http://www.nltk.org/api/nltk.tokenize.html

sentimentclassification approach[C]//Proceedings of the 20th ACM international conference on Information and knowledge management. ACM, 2011: 1031-1040.

[3]Nguyen D, Gravel R, Trieschnigg D, et al. " How Old Do You Think I Am?" A Study of Language and Age in Twitter[C]//ICWSM. 2013.

[4] Zamberletti A, Noce L, Gallo I. Text localization based on fast feature pyramids and multi-resolution maximally stable extremal regions[C]//Asian Conference on Computer Vision. Springer, Cham, 2014: 91-105.

[5] D. Estival, T. Gaustad, S.B. Pham, W. Radford, B. Hutchinson, Author profiling for English emails, in: Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING), 2007, pp. 262–272.

[6] Pak A, Paroubek P. Twitter as a corpus for sentiment analysis and opinion mining[C]//LREc. 2010, 10(2010).

[7] Term Frequency and Inverse Document Frequency (tf-idf) Using Tidy Data Principles 2018-03-21https://cran.r-project.org/web/packages/tidytext/vignettes/tf_idf.html

[8] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[C]//Advances in neural information processing systems. 2014: 3104-3112.

[9] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014.

[1]dataset source :https://modules.lancaster.ac.uk/pluginfile.php/1598262/mod_page/content/13/celebs.zip
[2] explanation quote from :http://www.nltk.org/api/nltk.tokenize.html