Twitter hashtag implication using various machine learning methods

Chang Wang
Department of Computer Sciences
wych92@cs.wisc.edu

Lichao Yin
Department of Computer Sciences
lichao@cs.wisc.edu

Chaowen Yu
Department of Computer Sciences
ycw@cs.wisc.edu

Biao Zhang
Department of Computer Sciences
bzhang263@wisc.edu

May 9, 2015

Abstract

keywords: supervised learning, text classification

1 Introduction

In this ever-changing world, information is produced and consumed ever faster than we could imagine. Every second, Twitter users post 6000 tweets, which corresponds to 350,000 tweets per minute, and 500 million tweets per day. With such amount of information generated, it can be formidable to search for a specific piece of information, if not completely impossible. That is why hashtag (#) comes into being. Formally, hashtag (#) is a character string that indicates the topic or category of the tweet. A tweet can have more than one hashtag and thus related to multiple topics. Hashtag greatly facilitates information filtering and rearrangements as searching for a tweet can be simplified to searching for a hashtag or even multiple hashtags.

However, not every user is used to hashtagging a tweet: they may forget to hashtag a tweet or they may use rare hashtags that could barely be keyword in searching or filtering. On the other hand, manually fixing a million tweets seems impractical even if tagless or bad-tagged tweets take up only 0.2% of all tweets posted every day.

Our project aims to solve this problem using various machine learning methods. Given the text of a tweet, our task is to imply an appropriate hashtag that indicates the category or topic of this tweet. Formally, we define the hashtag implication problem as a supervised learning classification model as follows.

Definition 1 (Hashtag Implication Problem)**.** *Given a S set of candidate hashtags as class labels, and a set T of tagged tweets as training set. Learn a model H that predicts one hashtag from S for an untagged tweet with high accuracy.*

We do not specify deliberately which machine learning model we will be using as long as it is supervised learning. This is because it is not clear at this moment which model has the best performance. We feel it is wise to explore various machine learning methods, compare their performances and find out which machine learning method is the most accurate on hashtag implication problem.

The rest of our report is structured as follows: Section 2 summarizes how we obtain training and testing data using Twitter API. Section 3 introduces implementation details for various machine learning methods including decision tree, neural network, support vector machine, etc. Performances of these methods are compared and analyzed in Section 4. Finally

Section 5 concludes this paper with some possible future work.

2 Data Preparation

2.1 Hashtag Selection

Unlike many may conceive, topic selection is no easy task. To select a set of hashtag candidates, we consider several factors. First, a hashtag should have a unique textual form. If one topic can have multiple hashtag forms, it may not possible to classify a set of tweets because a tweet may be put into any hashtag class under the same topic. Effective classification is depending on disjoint hashtag classes thus disjoint topics. For example `#butterfly` and `#butterflies` have exactly the same meaning and may be used as hashtags interchangeably, and having a class for `#butterfly` and another for `#butterfiles` does not only undermine our classification, but also conflicts with our project initiatives.

Second, a hashtag should have enough tweets for training. Too few tweets associated with a hashtag cannot fully characterize the hashtag and may not achieve effective training process. We think 1000 tweets for a hashtag are sufficient for training and testing. In reality, multiple hashtags are also possible. In this scenario we simply treat other hashtags as regular words and leave the only hashtag selected in our candidate set.

Combining these factors, we selected five hashtags, each having 1000 associated tweets. They are

- `#badgers`
- `#baltimoreuprising`
- `#gameofthrones`
- `#love`
- `#taylorswift`

2.2 Tweet Fetching

To facilitate publich usage and research, Twitter has publicized various APIs via which interested users can retrieve information from Twitter web server. There are two sets of Twitter APIs that we use. First, OAuth APIs are used to send secured authorized requests. We need to provide authentication and authorization information whenever we want to use Twitter API. This includes applying for a new application called Hashtag Prefetcher on the user profile and get a pair of Consumer Key and Consumer Secret for application and a pair of Access Key and Access Secret for user account. The second part, once authorized, is to make queries to server using the search API. The search API is part of Twitter REST API that return tweets satisfying a certain set of conditions such as posted time, hashtag, keyword, language, retweet count, etc. It is written in JSON and returns a series of Tweet objects.

In our project, we use a simple Twitter API wrapper called

2.3 Data Preprocessing

After above steps, we get 5000 tweets with corresponding hashtags like this:

```
   The House of Black and White -
The Wars to Come - #gameofthrones
#got #got5 #e2s5 #hbo \u2026
https://t.co/6zrQXilrZe
```

It is obvious that `http://`, `\u2026` and `#gameofthrones` do little contribution to classification. So we find them out through 5000 tweets and remove them. For convenience, we then remove all non-alphanumeric characters and turn all letters to lowercase. Thus the example tweet turns to be:

```
   the house of black and white the
wars to come
```

Because computer usually fails to capture semantic information from human languages, it is common to represent a text as a vector corresponding to the terms(usually terms are words) that appear in the text. For our problem, we will use bag-of-words model. Here we define the following terms as denoted as [1]:

- A *term* is the basic unit of discrete data denoted by w, which is usually a word appearing in a tweet.
- A *tweet* contains N terms, denoted by $\mathbf{w} = (w_1, w_2, \ldots, w_N)$.
- A *corpus* is a collection of M tweets($M = 5000$). The tweets texts are denoted by $\mathbf{D} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M\}$, and their corresponding tag labels are denoted by $\mathbf{Y} = \{y_1, y_2, \ldots, y_M\}$ where $y_i \in \{1, 2, 3, 4, 5\}, i = 1, 2, \ldots, M$. For our problem, 1 stands for the hashtag `badgers`, 2 stands for the hashtag `baltimoreuprising`, 3 stands for the hashtag `gameofthrones`, 4 stands for the hashtag `love`, and 5 stands for the hashtag `taylorswift`.
- A *dictionary* is a collection of all V unique terms appearing in the corpus denoted by $\mathbf{W} = \{w_1^*, w_2^*, \ldots, w_V^*\}$.

Let $\mathbf{X}$ be a term-tweet matrix. This matrix has M rows(one row for each tweets) and V columns(one column for each unique term in dictionary). The element x_{ij} in $\mathbf{X}$ indicates whether the j-th term appears in the i-th tweet. If it does appear, $x_{ij} = 1$; otherwise $x_{ij} = 0$.

In general, because most tweets will just use a small part of terms in the dictionary, a lot of x_{ij} will be zero. So the matrix $\mathbf{X}$ is sparse. However, a sparse $\mathbf{X}$ wastes too much space and processes much slower in high dimension. As we know, there are some high-frequency terms appearing in all documents, such as function words(e.g., *a*, *or*, *on*) and pronouns(e.g., *which*, *it*, *those*). These terms have little contribution to representing different documents because of relatively low information content, so we shall remove them from our dictionary. [2] developed a SMART system with a popular list of more than 500 common terms, which called *stop-word list* and can be used for our model.

After removing *stop-words*, our final $\mathbf{X}$ has 5000 rows for tweets and 9791 columns for unique terms, and $\mathbf{Y}$ has 5000 rows for each tweet's tag label.

2.4 Cross Validation

We will use stratified sampling for cross validation.

Since we have 1000 tweets for every 5 hashtags. We equally split each 1000 hashtag tweets into 5 parts. And each time we select and combine 4 parts from every hashtags for training and the rest part for testing.

Therefore, we have 5 datasets in all. Each dataset has training matricies: $\mathbf{X}_{training} \in \mathbb{R}^{4000 \times 9791}$ and $\mathbf{Y}_{training} \in \{1, 2, 3, 4, 5\}^{4000}$, and testing matrices $\mathbf{X}_{testing} \in \mathbb{R}^{1000 \times 9791}$ and $\mathbf{Y}_{testing} \in \{1, 2, 3, 4, 5\}^{1000}$.

3 Implementation

3.1 Decision Tree

Decision tree is a simple but powerful machine-learning model. The advantages of decision trees are: 1. Easy to interpret and explain. 2. Non-parametric, so you don't have to worry about outliers or whether the data is linearly separable. The disadvantage is also obvious that they easily overfit.

To learn a multi-class decision tree, we use the implementation from scikit-learn 1.6.1. scikit-learn uses an optimised version of the CART algorithm.

We use scikit-learn default configuration to finish our task. The default configurations are as follows:

- *criterion*: Gini impurity function is used to measure the quality of a split.
- *splitter strategy*: choose the best at when splitting.
- *number of features to consider when looking for the best split*: consider max number of features at each split since all features are of integer type.
- *maximum depth of the tree*: nodes are expanded until all leaves are pure or until all leaves contain less than a threshold.
- *minimum number of samples for splitting*: The minimum number of samples required to split an internal node. Set to 2.
- *minimum number of samples for a leaf node*

: The minimum number of samples required to be at a leaf node. Set to 1.

3.2 Random Forest

Random Forest is an ensemble method that produces a highly accurate classifier and learns fast. It runs efficiently on large data bases and handles large amount of input variables without variable deletion or selection.

We download an Matlab implementation code from *https://code.google.com/p/randomforest-matlab*. For our problem we set the number of trees is 500 and the maximum trial times is 98.

3.3 Neural Network

Neural networks are algorithms that can be used to perform nonlinear statistical modeling and provide a new alternative to logistic regression, the most commonly used method for developing predictive models. Neural network requires less formal statistical training and implicitly detects complex nonlinear relationships between dependent and independent variables.

For our classification problem, we use neural network toolbox in matlab. We just use one hidden layer with 10 neurons and the other configurations are as follows:

- *Data Division*: Random (dividerand)
- *Training*: Scaled Conjugate Gradient (trainscg)
- *Performance*: Cross-Entropy (crossentropy)
- *Derivative*: Default (defaultderiv)

3.4 Naïve Bayes

The naïve Bayes algorithm is one of the two classic naïve Bayes variants used in text classification where the data are typically represented as word vector counts. The biggest advantages of naïve Bayes is that it is super simple. Furthermore, sometimes even if the naïve Bayes assumption doesn't hold, a naïve Bayes classifier still often performs surprisingly well in practice.

The implementation of naïve Bayes we used is the multinomial naïve Bayes MultinomialNB from scikit-learn 1.6.1.

The configuration we used in our project is the default one, which are as follows:

- *additive (Laplace/Lidstone) smoothing parameter* : 1.0
- *whether to learn class prior probabilities or not*: True
- *prior probabilities of the classes*: None

3.5 Support Vector Machine

Advantages of SVMs: High accuracy, nice theoretical guarantees regarding overfitting, and with an appropriate kernel they can work well even if you're data isn't linearly separable in the base feature space. Especially popular in text classification problems where very high-dimensional spaces are the norm. Memory-intensive and kind of annoying to run and tune, though, so I think random forests are starting to steal the crown.

SVM uses kernel trick to map inputs to higher dimension so that they can be linear classified.

Parameters setting:

- *kernel function:* Gaussian function
- *sigma:* number of features

3.6 Logistic Regression Model

Logistic Regression can deal with binomial or multinomial classification problems. It can be seen a simplified version of sigmoid neural network without hidden layer. The result can be seen the confidence measure of every category.

Here we used LIBLINEAR to train and test the tweet data.

3.7 k-Nearest Neighbors

The k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method. So we don't have to tune the model parameters. As an instanced-based algorithm where every prediction is dependent on local information, it is very sensitive to local structure.

We used Matlab built-in kNN implementation `fitcknn` with $k = 1$ an $k = 5$.

4 Results

In detail, we find something interesting to address. Even if naïve Bayes is not the implementing algorithm with best performance, it indeed draws some outstanding predictions in some cases, which shows its availability and accuracy. For example, all other implementing algorithms predicts the following tweet as `love` and only naïve Bayes predicts it as `badgers`.

```
I may have cried when illustrating @AngelaSlatter's "Spells for Coming Forth By Daylight" http://t.co/9IMhTRLj49 books badgers endings
```

Another thing that shall be addressed is that our dataset does have some noise. In our first dataset, we find that all implementing algorithms predict wrong for the 528th testing tweet. It shall belongs to the hashtag `gameofthrones`, but several algorithms treat it as `love`, `badgers` or `baltimoreuprising`. That's because the original content of the tweet is too simple and "common", which is

```
Yep... we've all been there. \ud83d\ude29 http://t.co/XTucN2D4yx #ILoveMondays #Sarcasm #GameofThrones #GoT #Joffrey
```

It is obvious that this tweet contains no outstanding semantic terms which indicates `gameofthrones` meaning. And this case is not unique in our dataset, which affects the overall performance.

5 Conclusion

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[2] Gerard Salton and M McGill. The smart retrieval systemexperiments in automatic document retrieval, 1971.

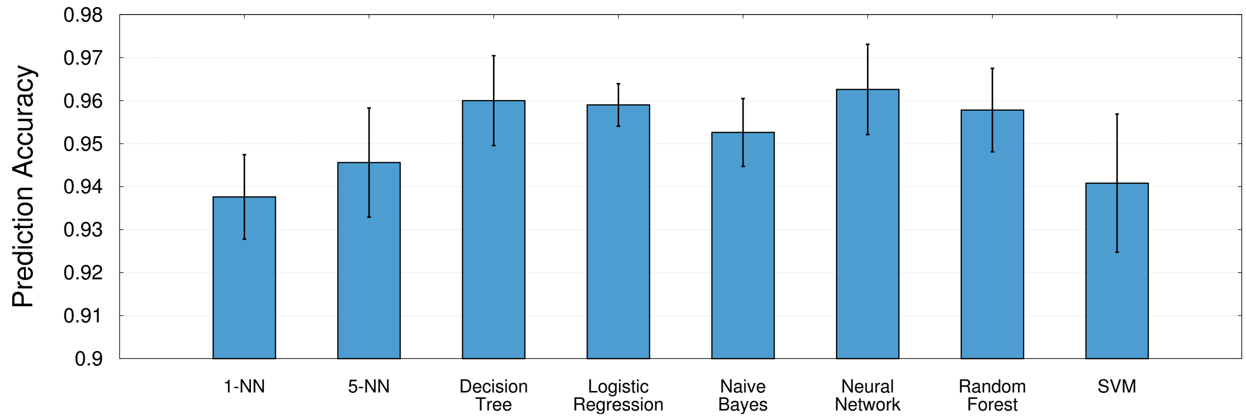

Figure 1: Boot-up time for various configurations

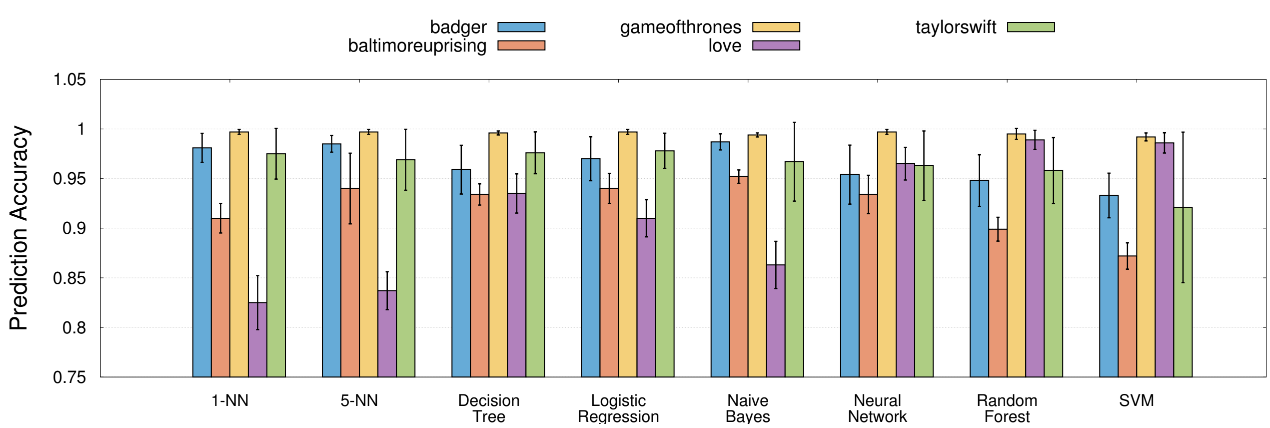

Figure 2: Boot-up time for various configurations