Twitter Hash Tag Prediction Algorithm

Twitter is one popular web application nowadays. Twitter allows users to use "Hash tags" to classify their tweets. In this research project, we propose an algorithm to predict tags, by utilizing machine learning and network relatedness methods.

1. Problem Definition

Hash tag prediction is different from normal texts classification. Here we don't know how many clusters we need to find. In addition, the tag set changes so frequently that it is almost impossible to effectively carry out classification or clustering, since a new tag would force us to establish a new class and a new classification rule. Our intuition is: if we can measure the correlation between various tweets as the mathematical metric we can treat the collected tweets as points in a high dimensional space, and construct a network by the latent space model.

2. Method

2.1 Theory

An intuitive way to solve this problem is to use Euclidean distance between points as the measurement of their similarity. We developed our theory based on this distance. Since in a Euclidean Space, the distance is equivalent to the norm of a vector, we will focus our discussion on norms.

Let $\mathbf{u}_1, \mathbf{u}_2 \cdots \mathbf{u}_p$ be the standard bases (with unit norm) of a *p*-dimensional Euclidean Space. Then for any vector $\mathbf{v}$ with coordinates $\begin{pmatrix} x_1 & x_2 & \cdots & x_{p-1} & x_p \end{pmatrix}$, we have $\mathbf{v} = \sum_{i=1}^{p} x_i \mathbf{u}_i$. Then the Euclidean norm of vector $\mathbf{v}$ is given by

$$\|\mathbf{v}\|^2 = \mathbf{v} \cdot \mathbf{v} = \sum_{i=1}^{p} x_i \mathbf{u}_i \cdot \sum_{i=1}^{p} x_i \mathbf{u}_i = \sum_{i,j=1}^{p} x_i x_j \mathbf{u}_i \cdot \mathbf{u}_j \qquad (1)$$

where $\cdot$ represents the inner product operation defined in the Euclidean Space. Clearly, if we assume $\mathbf{u}_i \cdot \mathbf{u}_j = 0$, that is, $\mathbf{u}_i$ and $\mathbf{u}_j$ are orthogonal, whenever $i \neq j$, the Euclidean norm equals to $\|\mathbf{v}\|^2 = \sum x_i^2$. In our problem, the bases are the words in the dictionary. The preliminary assumption for Euclidean distance is that the bases are orthogonal to each other, that is, the words in dictionary are uncorrelated, which is against common sense. Therefore, we need to perform some transformation to capture this correlation.

In formula (1), as $\mathbf{u}_i$ and $\mathbf{u}_j$ are unit vectors, their inner product is actually the

cosine of the angle between them. Thus we can rewrite (1) in a matrix form as

$$\|\mathbf{v}\|^2 = \begin{pmatrix} x_1 & \cdots & x_p \end{pmatrix} \begin{pmatrix} \cos\theta_{11} & \cdots & \cos\theta_{1p} \\ \vdots & \ddots & \vdots \\ \cos\theta_{p1} & \cdots & \cos\theta_{pp} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} = XMX^T \qquad (2)$$

where $\cos\theta_{ii} = 1, i = 1 \cdots p$ and $X = \begin{pmatrix} x_1 & x_2 & \cdots & x_{p-1} & x_p \end{pmatrix}$.

Now we try to find the angle between each pair of terms in the dictionary and then calculate the matrix M. Notice that M is clearly a symmetric and non-negative definite matrix. If we decompose M in the way

$$M = CC^T \qquad (3),$$

then (2) becomes $\|\mathbf{v}\|^2 = XCC^T X^T = \tilde{X}\tilde{X}^T$ where $\tilde{X} = XC$. So the norm can be seen as the Euclidean norm of the transformed coordinates. Here we take (3) as the Eigen value decomposition of M, so $\tilde{X}$ could be the coordinates of vector $\mathbf{v}$ in a new coordinate system where axes are orthogonal to each other. Please note that we can use any other decomposition in the form of (3) to get the same norm in computation, even when C is not a square matrix. With this property, the computation becomes applicable.

2.2 Estimate the Cosine Matrix

First, we construct the preliminary weighted matrix, say H, by using the WordNet to initialize the semantic correlation among words from the dictionary. If two words t_i, t_j are similar to each other, and they both appear in one Tweet, we add positive weights for both words. This process can be expressed as

$$\hat{x}_i = x_i + \sum_{j \neq i}^{p} \rho_{ij} x_j$$

where $\rho_{ij} \in (0,1)$, equals to one when t_i, t_j are similar words and zero otherwise. Here we take the same positive number ρ for all $\rho_{ij} \in (0,1)$, and if $\rho_{ij} > 0$, so is ρ_{ji}. Then we can construct the symmetric matrix H as

$$\mathrm{H} = \begin{pmatrix} 1 & \cdots & \rho_{1p} \\ \vdots & \ddots & \vdots \\ \rho_{p1} & \cdots & 1 \end{pmatrix} \qquad \hat{X} = X\mathrm{H} \qquad (4)$$

In the second step, we get m tweets, say $X_1 \cdots X_m$, and transform them by (4) to get $\hat{X}_1 \cdots \hat{X}_m$. Then by these data, we use cosine similarity in variable analysis to

construct matrix M . Set the text matrix as the $m \times p$ matrix

$$\Omega = (\hat{X}_1^T \quad \cdots \quad \hat{X}_m^T)^T = \begin{pmatrix} \hat{x}_{11} & \cdots & \hat{x}_{1p} \\ \vdots & \ddots & \vdots \\ \hat{x}_{m1} & \cdots & \hat{x}_{mp} \end{pmatrix}.$$

We would estimate the cosine between the i'th and j'th terms as

$$\cos\theta_{ij} = \hat{X}_{\bullet i}^T \hat{X}_{\bullet j} \Big/ \left\| \hat{X}_{\bullet i} \right\| \left\| \hat{X}_{\bullet j} \right\| = \frac{\sum_{k=1}^{m} \hat{x}_{ki} \hat{x}_{kj}}{\sqrt{\sum_{k=1}^{m} \hat{x}_{ki}^2 \sum_{k=1}^{m} \hat{x}_{kj}^2}} \qquad (5)$$

The distance estimate obtained from formula (5) is equivalent to what proposed by *L. Jing, L. Zhou, K. Ng and J. Huang* (2006), but with a better mathematical explanation. Note that since our data is represented as frequency, all the elements of the matrix Ω would be non-negative. So the cosine estimated in this way can only be non-negative. Therefore, all angles between words are cute or right angles. In this way, all words tend to be similar to each other in some degree. This may well incorporate the similarity elements, but might also be vulnerable to noise. In the following, we give a modified estimate which also includes the possibility of obtuse angle and takes dissimilarity into consideration, which is also the sample correlation in statistics,

$$\cos\theta_{ij} = \frac{\sum_{k=1}^{m} (\hat{x}_{ki} - \hat{x}_{\bullet i})(\hat{x}_{kj} - \hat{x}_{\bullet j})}{\sqrt{\sum_{k=1}^{m} (\hat{x}_{ki} - \hat{x}_{\bullet i})^2 \sum_{k=1}^{m} (\hat{x}_{kj} - \hat{x}_{\bullet j})^2}} \qquad (6)$$

Since the distance from (5) was named as Ontology Based Distance (OBD) in the original paper, here we call the distance in (6) centralized Ontology Based Distance (COBD). We will discuss the pros and cons of the two methods in Section 3. In the following sub section, we will make another adjustment to the method.

2.3 Normalization

Note that the various scales of vectors may still cause us some problem. Consider a special case where $\tilde{X}_1 = (1,0,0)$, $\tilde{X}_2 = (10,0,0)$, $\tilde{X}_3 = (0,0,1)$. Obviously, $\tilde{X}_1$ and $\tilde{X}_2$ here should have high similarity value between them. But in this case, the distance between $\tilde{X}_1$ and $\tilde{X}_3$ is much smaller.

To make our method more reasonable, before we compute the distance between transformed points, we need to rescale their distances to the original point as 1. And then we measure the Euclidean distance between normalized points.

2.4 Prediction of Tags

Finally, we try to predict tags based on the distance. An intuitive way is to simply select the tag of the closest tweet. In this case, it may be unwise to simply pick the closest tweet's tag, since that relies on the acurracy of distance too heavily. To increase the accuracy, we collect a few closest tweets, and make the prediction based on tag ratios. Specifically, we will collect *n* initial closest tweets at first. Then from this point, we will keep adding tweets while check a certain tag has become dominate. If there is a tag with a ratio higher than 50%, we will choose this tag as our primary predicted tag. Since in some cases tags have very similar meanings (such as #government vs. #election), sometimes we will also pick a secondary tag to predict.

3. Results and Discussion

To compare the performances of various distances discussed above, we use a test dataset consist of 400 tweets that are not included in the sample set we used to estimate matrix M . There are 4 different tags. We first process the OBD on a dataset with 665 tweets that are not in our test set, choose the best performance ρ (=0.2) and use it for both OBD and COBD. The table below shows the test result for Euclidean Distance (EucD), OBD and COBD.

	Test Error Rate	Type II Error
EucD	16.25%	5.1%
COBD	13.5%	4.6%
OBD	12.75%	4.2%

Table1: The test error rate and type II error for three distances. Type II error is the rate we assign a wrong tag to a particular tweet.

Both OBD and COBD outperform EucD, and OBD is the best one. If we see the data for different tags (not provided here for concise), we would find COBD is the most stable one, while EucD is far more unstable. But the disadvantage of COBD lies in computation. We need to estimate the cosine matrix M to construct the distance, which involves computation for matrices with tens of thousands rows and columns. It won't be a big problem for OBD since the matrices are sparse. But in COBD, the matrix becomes non-sparse, so we need many decompositions and transformations of matrices to make the computation applicable. Given their close performances, OBD is more practical in application, while the COBD is a better model theoretically.

The left picture in Figure 1 shows the COBD from other tweets to a random selected tweet. Different colors represent tweets with different tags. It can be seen that most of the tweets are very close to the 1.4142 distance boundary, and the majority of points falling in the circle are from the correct tag group. This indicates that tweets with different topics are projected onto orthogonal axes. The right plot illustrates the distance distribution. The lighter the color is, the shorter the corresponding distance is. Since the tweets are sorted by tags, we see that the distance within each group appears

to be shorter, as shown by the light rectangles along the diagonal.

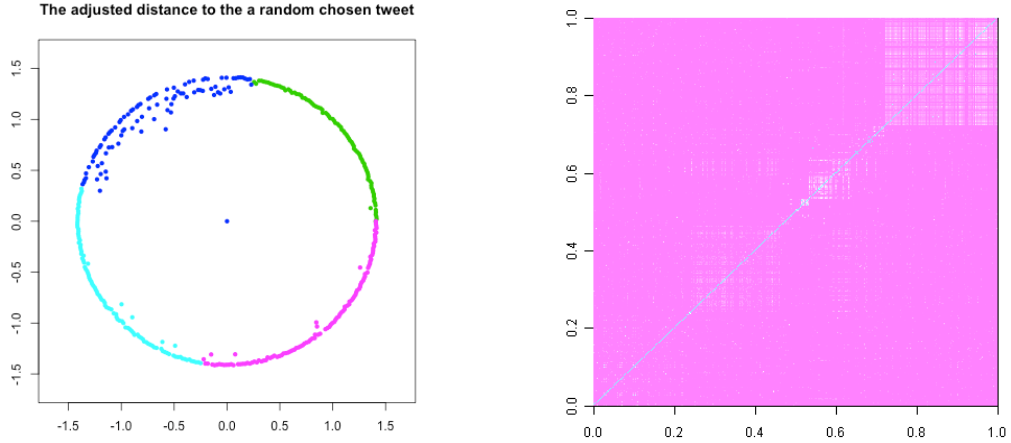

Figure 1: the distance to one point and the distribution of sample distance matrix

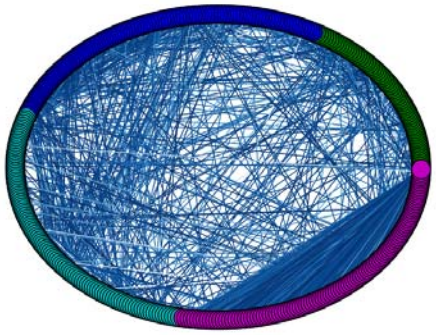

Figure 2: Prediction Visualization

In Figure2, different colors represent what tag cluster the tweets belong to. A link will be added between a pair of nodes when they are near enough. In addition, the deeper color the line is, the higher the similarity value is. As we can see, the lines appear to be very dense among each tag cluster, and sparse between tweets with different tags. It indicates that tweets with the same tag cluster are near on average.

Due to the vagueness of many tweets, the correct rate of more than 86% is actually very high. Apart from the accuracy, our method has other advantages:

(1) The whole system is easy to store (we only need to store the *C* matrix in (3)).
(2) It is easy to update when dictionary changes (only needs to compute an extra column and add it back to original matrix)
(3) It won't lose power when the topics trend changes with time, and it can work with personal elements and settings, which makes it more flexible (since we can set the algorithm to only consider the distance of the objective tweet to certain subset of other tweets, so elements like location, time, etc can be incorporated.)
(4) In addition, the distance provides us the possibility to transform the twitter system and even other text systems into social networks by latent space approach. So we can use traditional social network methods to discuss the properties of such systems.