Fast Tweet Retrieval with Compact Binary Codes

Weiwei Guo* **Wei Liu†** **Mona Diab‡**
*Computer Science Department, Columbia University, New York, NY, USA
†IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
‡Department of Computer Science, George Washington University, Washington, D.C., USA
weiwei@cs.columbia.edu weiliu@us.ibm.com mtdiab@gwu.edu

Abstract

The most widely used similarity measure in the field of natural language processing may be cosine similarity. However, in the context of Twitter, the large scale of massive tweet data inevitably makes it expensive to perform cosine similarity computations among tremendous data samples. In this paper, we exploit binary coding to tackle the scalability issue, which compresses each data sample into a compact binary code and hence enables highly efficient similarity computations via Hamming distances between the generated codes. In order to yield semantics sensitive binary codes for tweet data, we design a binarized matrix factorization model and further improve it in two aspects. First, we force the projection directions employed by the model nearly orthogonal to reduce the redundant information in their resulting binary bits. Second, we leverage the tweets' neighborhood information to encourage similar tweets to have adjacent binary codes. Evaluated on a tweet dataset using hashtags to create gold labels in an information retrieval scenario, our proposed model shows significant performance gains over competing methods.

1 Introduction

Twitter is rapidly gaining worldwide popularity, with 500 million active users generating more than 340 million tweets daily[1]. Massive-scale tweet data which is freely available on the Web contains rich linguistic phenomena and valuable information, therefore making it one of most favorite data sources used by a variety of Natural Language Processing (NLP) applications. Successful examples include first story detection (Petrovic et al., 2010), local event detection (Agarwal et al., 2012), Twitter event discovery (Benson et al., 2011) and summarization (Chakrabarti and Punera, 2011), etc.

In these NLP applications, one of core technical components is tweet similarity computing to search for the desired tweets with respect to some sample tweets. For example, in first story detection (Petrovic et al., 2010), the purpose is to find an incoming tweet that is expected to report a novel event not revealed by the previous tweets. This is done by measuring *cosine* similarity between the incoming tweet and each previous tweet.

One obvious issue is that cosine similarity computations among tweet data will become very slow once the scale of tweet data grows drastically. In this paper, we investigate the problem of searching for most similar tweets given a query tweet. Specifically, we propose a binary coding approach to render computationally efficient tweet comparisons that should benefit practical NLP applications, especially in the face of massive data scenarios. Using the proposed approach, each tweet is compressed into short-length binary bits (*i.e.*, a *compact binary code*), so that tweet comparisons can be performed substantially faster through measuring Hamming distances between the generated compact codes. Crucially, Hamming distance computation only involves very cheap NOR and popcount operations instead of floating-point operations needed by cosine similarity computation.

Compared to other genres of data, similarity search in tweet data is very challenging due to the short nature of Twitter messages, that is, a tweet contains too little information for traditional models to extract

[1] http://en.wikipedia.org/wiki/Twitter

Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 486–496, Dublin, Ireland, August 23-29 2014.

Symbol	Definition
n	Number of tweets in the corpus.
d	Dimension of a tweet vector, *i.e.*, the vocabulary size.
$\boldsymbol{x}_i$	The sparse *tf-idf* vector corresponding to the i-th tweet in the corpus.
$\bar{\boldsymbol{x}}_i$	The vector subtracted by the mean $\boldsymbol{\mu}$ of the tweet corpus: $\bar{\boldsymbol{x}}_i = \boldsymbol{x}_i - \boldsymbol{\mu}$.
$X, \bar{X}$	The tweet corpus in a matrix format, and the zero-centered tweet data.
r	The number of binary coding functions, *i.e.*, the number of latent topics.
f_k	The k-th binary coding function.

Table 1: Symbols used in binary coding.

latent topical semantics. For instance, in our collected dataset, there exist only 11 words per tweet on average. We address the sparsity issue pertaining to tweet data by converting our previously proposed topic model*Weighted Textual Matrix Factorization* (WTMF) (Guo and Diab, 2012) to a binarized version. WTMF maps a tweet to a low-dimensional semantic vector which can easily be transformed to a binary code by virtue of a sign function. We consider WTMF a good baseline for the task of tweet retrieval, as it has achieved state-of-the-art performance among unsupervised systems on two benchmark short-text datasets released by Li et al. (2006) and Agirre et al. (2012).

In this paper, we improve WTMF in two aspects. The first drawback of the WTMF model is that it focuses on exhaustively encoding the local context, and hence introduces some overlapping information that is reflected in its associated projections. In order to remove the redundant information and meanwhile discover more distinct topics, we employ a gradient descent method to make the projection directions nearly orthogonal.

The second aspect is to enrich each tweet by its neighbors. Because of the short context, most tweets do not contain sufficient information of an event, as noticed by previous work (Agarwal et al., 2012; Guo et al., 2013). Ideally, we would like to learn a model such that the tweets related to the same event are mapped to adjacent binary codes. We fulfill this purpose by augmenting each tweet in a given training dataset with its neighboring tweets within a temporal window, and assuming that these neighboring (or similar) tweets are triggered by the same event. We name the improved model *Orthogonal Matrix Factorization with Neighbors* (OrMFN).

In our experiments, we use Twitter hashtags to create the gold (*i.e.*, groundtruth) labels, where tweets with the same hashtag are considered semantically related, hence relevant. We collect a tweet dataset which consists of 1.35 million tweets over 3 months where each tweet has exactly one hashtag. The experimental results show that our proposed model OrMFN significantly outperforms competing binary coding methods.

2 Background and Related Work

2.1 Preliminaries

We first introduce some notations used in this paper to formulate our problem. Suppose that we are given a dataset of n tweets and the size of the vocabulary is d. A tweet is represented by all the words it contains. We use notation $\boldsymbol{x} \in \mathbb{R}^d$ to denote a sparse d-dimensional *tf-idf* vector corresponding to a tweet, where each word stands for a dimension. For ease of notation, we represent all n tweets in a matrix $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$. For binary coding, we seek r binarization functions $\left\{ f_k : \mathbb{R}^d \rightarrow \{1, -1\} \right\}_{k=1}^{r}$ so that a tweet $\boldsymbol{x}_i$ is encoded into an r-bit binary code (*i.e.*, a string of r binary bits). Table 1 illustrates the symbols used in this paper for notation.

Hamming Ranking: In the paper we evaluate the quality of binary codes in terms of Hamming ranking. Given a query tweet, all data items are ranked in an ascending order according to the Hamming distances between their binary codes and the query's binary code, where a Hamming distance is the number of bit positions in which bits of two codes differ. Compared with cosine similarity, computing Hamming distance can be substantially efficient. This is because fixed-length binary bits enable very cheap logic operations for Hamming distance computation, whereas real-valued vectors require floating-point op-

erations for cosine similarity computation. Since logic operations are much faster than floating-point operations, Hamming distance computation is typically much faster than cosine similarity computation[2]

2.2 Binary Coding

Early explorations of binary coding focused on using random permutations or random projections to obtain binary coding functions (aka, hash functions), such as Min-wise Hashing (MinHash) (Broder et al., 1998) and Locality-Sensitive Hashing (LSH) (Indyk and Motwani, 1998). MinHash and LSH are generally considered *data-independent* approaches, as their coding functions are generated in a randomized fashion. In the context of Twitter, the simple LSH scheme proposed in (Charikar, 2002) is of particular interest. Charikar proved that the probability of two data points colliding is proportional to the angle between them, and then employed a random projection $\boldsymbol{w} \in \mathbb{R}^d$ to construct a binary coding function:

$$f(\boldsymbol{x}) = \text{sgn}\big(\boldsymbol{w}^\top \boldsymbol{x}\big) = \begin{cases} 1, & \text{if } \boldsymbol{w}^\top \boldsymbol{x} > 0, \\ -1, & \text{otherwise.} \end{cases} \quad (1)$$

The current held view is that *data-dependent* binary coding can lead to better performance. A data-dependent coding scheme typically includes two steps: 1) learning a series of binary coding functions with a small amount of training data; 2) applying the learned functions to larger scale data to produce binary codes.

In the context of tweet data, Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997) can directly be used for data-dependent binary coding. LSA reduces the dimensionality of the data in X by performing singular value decomposition (SVD) over X: $X = U\Sigma V^\top$. Let $\bar{X}$ be the zero-centered data matrix, where each tweet vector $\boldsymbol{x}_i$ is subtracted by the mean vector $\boldsymbol{\mu}$, resulting in $\bar{\boldsymbol{x}}_i = \boldsymbol{x}_i - \boldsymbol{\mu}$. The r coding functions are then constructed by using the r eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_r$ associated with the r largest eigenvalues, that is, $f_k(\boldsymbol{x}) = \text{sgn}\big(\boldsymbol{u}_k^\top \bar{\boldsymbol{x}}\big) = \text{sgn}\big(\boldsymbol{u}_k^\top (\boldsymbol{x} - \boldsymbol{\mu})\big)$ $(k = 1, \cdots, r)$. The goal of using zero-centered data $\bar{X}$ is to balance 1 bits and -1 bits.

Iterative Quantization (ITQ) (Gong and Lazebnik, 2011) is another popular unsupervised binary coding approach. ITQ attempts to find an orthogonal rotation matrix $R \in \mathbb{R}^{r \times r}$ to minimize the squared quantization error: $\|B - RV\|_{\text{F}}^2$, where $B \in \{1, -1\}^{r \times n}$ contains the binary codes of all data, $V \in \mathbb{R}^{r \times n}$ contains the LSA-projected and zero-centered vectors, and $\|\cdot\|_{\text{F}}$ denotes Frobenius norm. After R is optimized, the binary codes are simply obtained by $B = \text{sgn}(RV)$.

Much recent work learns nonlinear binary coding functions, including Spectral Hashing (Weiss et al., 2008), Anchor Graph Hashing (Liu et al., 2011), Bilinear Hashing (Liu et al., 2012b), Kernelized LSH (Kulis and Grauman, 2012), etc. Concurrently, supervised information defined among training data samples was incorporated into coding function learning such as Minimal Loss Hashing (Norouzi and Fleet, 2011) and Kernel-Based Supervised Hashing (Liu et al., 2012a). Our proposed method falls into the category of *unsupervised, linear, data-dependent* binary coding.

2.3 Applications in NLP

The NLP community has successfully applied LSH in several tasks such as first story detection (Petrovic et al., 2010), and paraphrase retrieval for relation extraction (Bhagat and Ravichandran, 2008), etc. This paper shows that our proposed data-dependent binary coding approach is superior to data-independent LSH in terms of the quality of generated binary codes.

Subercaze et al. (2013) proposed a binary coding approach to encode user profiles for recommendations. Compared to (Subercaze et al., 2013) in which a data unit is a whole user profile consisting of all his/her Twitter posts, we tackle a more challenging problem, since our data units are extremely short – namely, a single tweet.

[2]We recognize that different hardware exploiting techniques such as GPU or parallelization accelerate cosine similarity. However, they don't change the inherent nature of the data representation. They can be equally applied to Hamming distance and we anticipate significant speed gains. We relegate this exploration of different implementations of Hamming distance to future work.

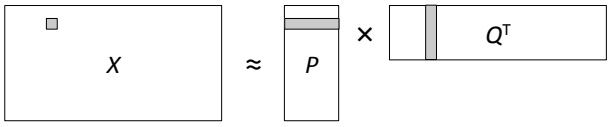


Figure 1: Weighted Textual Matrix Factorization. The $d \times n$ matrix X is approximated by the product of a $d \times r$ matrix P and an $n \times r$ matrix Q. Note in the figure we used the transpose of the Q matrix.

3 Weighted Textual Matrix Factorization

The WTMF model proposed by Guo and Diab (2012) is designed to extract latent semantic vectors for short textual data. The low-dimensional semantic vectors can be used to represent the tweets in the original high-dimensional space. WTMF achieved state-of-the-art unsupervised performance on two short text similarity datasets, which can be attributed to the fact that WTMF carefully handles missing words (the missing words of a text are the words with 0 values in a data vector $\boldsymbol{x}$).

Assume that there are r latent dimensions/topics in the data, the matrix X is approximated by the product of a $d \times r$ matrix P and an $n \times r$ matrix Q, as in Figure 1. Accordingly, a tweet $\boldsymbol{x}_j$ is represented by an r-dimensional vector $Q_{j,\cdot}$; similarly, a word w_i is generalized by the r-dimensional vector $P_{i,\cdot}$ (the ith row in matrix P). The matrix factorization scheme has an intuitive explanation: the inner-product of a word profile vector $P_{i,\cdot}$ and a tweet profile vector $Q_{j,\cdot}$ is to approximate the TF-IDF value X_{ij}: ${P_{i,\cdot}}^\top Q_{j,\cdot} \approx X_{ij}$ (as illustrated by the shaded parts in Figure 1).

Intuitively, $X_{ij} = 0$ suggests that the latent topics of the text $\boldsymbol{x}_j$ are not relevant to the word w_i. Note that 99% of the cells in X are 0 because of the short contexts, which significantly diminishes the contribution of the observed words to the searching of optimal P and Q. To reduce the impact of missing words, a small weight w_m is assigned to each 0 cell of X in the objective function:

$$\sum_i \sum_j W_{ij} \left({P_{i,\cdot}}^\top Q_{j,\cdot} - X_{ij} \right)^2 + \lambda ||P||_2^2 + \lambda ||Q||_2^2,$$
$$W_{i,j} = \begin{cases} 1, & \text{if } X_{ij} \neq 0, \\ w_m, & \text{if } X_{ij} = 0. \end{cases} \tag{2}$$

where λ is the regularization parameter. Alternating Least Squares (Srebro and Jaakkola, 2003) is used to iteratively compute the latent semantic vectors in P and Q:

$$P_{i,\cdot} = \left(Q^\top \tilde{W}^{(i)} Q + \lambda I \right)^{-1} Q^\top \tilde{W}^{(i)} X_{i,\cdot}^\top,$$
$$Q_{j,\cdot} = \left(P^\top \tilde{W}^{(j)} P + \lambda I \right)^{-1} P^\top \tilde{W}^{(j)} X_{\cdot,j} \tag{3}$$

where $\tilde{W}^{(i)} = \text{diag}(W_{i,\cdot})$ is a $n \times n$ diagonal matrix containing the i-th row of the weight matrix W. Similarly, $\tilde{W}^{(j)} = \text{diag}(W_{\cdot,j})$ is a $d \times d$ diagonal matrix containing the j-th column of W.

As in Algorithm 1 line 6-9, P and Q are computed iteratively, i.e., in a iteration each $P_{i,\cdot}(i = 1, \cdots, d)$ is calculated based on Q, then each $Q_{j,\cdot}(j = 1, \cdots, n)$ is calculated based on P. This can be computed efficiently since: (1) all $P_{i,\cdot}$ share the same $Q^\top Q$; similarly all $Q_{j,\cdot}$ share the same $P^\top P$; (2) X is very sparse. More details can be found in (Steck, 2010).

Adapting WTMF to binary coding is straightforward. Following LSA, we use the matrix P to linearly project tweets into low-dimensional vectors, and then apply the sign function. The k-th binarization function uses the k-th column of the P matrix ($P_{\cdot,k}$) as follows

$$f_k(\boldsymbol{x}) = \text{sgn}\left(P_{\cdot,k} \bar{\boldsymbol{x}}\right) = \begin{cases} 1, & \text{if } P_{\cdot,k} \bar{\boldsymbol{x}} > 0, \\ -1, & \text{otherwise.} \end{cases} \tag{4}$$

4 Removing Redundant Information

It is worth noting that there are two explanations of the $d \times r$ matrix P. The rows of P, denoted by $P_{i,\cdot}$, may be viewed as the collection of r-dimensional latent profiles of words, which we observe frequently

Algorithm 1: OrMF

Procedure P = `OrMF` $(X, W, \lambda, n\_itr, \alpha)$
$n\_words, n\_docs \leftarrow$ size(X);
randomly initialize P, Q;
$itr \leftarrow 1$;
while $itr < n\_itr$ **do**
  for $j \leftarrow 1$ **to** n_docs **do**
    $Q_{j,\cdot} = \left(P^\top \tilde{W}^{(j)} P + \lambda I\right)^{-1} P^\top \tilde{W}^{(j)} X_{\cdot,j}$
  for $i \leftarrow 1$ **to** n_words **do**
    $P_{i,\cdot} = \left(Q^\top \tilde{W}^{(i)} Q + \lambda I\right)^{-1} Q^\top \tilde{W}^{(i)} X_{i,\cdot}^\top$
  $c =$ mean(diag$(P^\top P)$);
  $P \leftarrow P - \alpha P(P^\top P - cI)$;
  $itr \leftarrow itr + 1$;

in the WTMF model. Meanwhile, columns of P are projection vectors, denoted by $P_{\cdot,k}$, which are similar to eigenvectors U obtained by LSA. The projection vector $P_{\cdot,k}$ is employed to multiply to a zero centered data vector $\bar{\boldsymbol{x}}$ to generate a binary string: sgn$({P_{\cdot,k}}^\top \bar{\boldsymbol{x}})$. In this section, we focus on the property of the P matrix columns.

As in equation 3, each row in matrices P and Q is iteratively optimized to approximate the data: ${P_{i,\cdot}}^\top Q_{j,\cdot} \approx X_{ij}$. While it does a good job at preserving the existence/relevance of each word in a short text, it might encode repetitive information by means of the dimensionality reduction or the projection vectors $P_{\cdot,k}$ (the columns of P). For example, the first dimension $P_{\cdot,1}$ may be 90% about the $politics$ topic and 10% about the $economics$ topic, and the second dimension $P_{\cdot,2}$ is 95% on $economics$ and 5% on $technology$ topics, respectively.

Ideally we would like the dimensions to be uncorrelated, so that more distinct topics of data could be captured. One way to ensure the uncorrelatedness is to force P to be orthogonal, i.e., $P^\top P = I$. It implies ${P_{\cdot,j}}^\top P_{\cdot,k} = 0$ if $k \neq j$.

4.1 Implementation of Orthogonal Projections

To produce nearly orthogonal projections in the current framework, we could add a regularizer $\beta(P^\top P - I)^2$ with the weight β in the objective function of the WTMF model (equation 6). However, in practice this method does not lead to the convergence of P. This is mainly caused by the phenomenon that any word profile $P_{i,\cdot}$ becomes dependent of all other word profiles after an iteration.

Therefore, we adopt a simpler method, gradient descent, in which P is updated by taking a small step in the direction of the negative gradient of $(P^\top P - I)^2$. It is also worth noting that $(P^\top P - I)^2$ requires each projection $P_{\cdot,k}$ to be a unit vector because of ${P_{\cdot,k}}^\top P_{\cdot,k} = 1$, which is infeasible when the nonzero values in X are large. Therefore, we multiply the matrix I by a coefficient c, which is calculated from the mean of the diagonal of $P^\top P$ in the current iteration. The following two lines are added at the end of an iteration:

$$\begin{aligned} c &\leftarrow \text{mean}(\text{diag}(P^\top P)), \\ P &\leftarrow P - \alpha P(P^\top P - cI). \end{aligned} \tag{5}$$

This procedure is presented in Algorithm 1. Accordingly, the magnitude of P is not affected. The step size α is fixed to 0.0001. We refer to this model as Orthogonal Matrix Factorization (OrMF).

5 Exploiting Nearest Neighbors for Tweets

We observe that tweets triggered by the same event do not have very high cosine similarity scores among them. This is caused by the inherent short length of tweets such that usually a tweet only describes one

aspect of an event (Agarwal et al., 2012; Guo et al., 2013). Our objective is to find the relevant tweets given a tweet, and then learn a model that assigns similar binary bits to these relevant tweets.

5.1 Modeling Neighboring Tweets

Given a tweet, we treat its nearest neighbors in a temporal window as its most relevant tweets. We assume that the other aspects of an event can be found in its nearest neighbors. Accordingly, we extract t neighbors for a tweet from 10,000 most chronologically close tweets. In this current implementation, we set $t = 5$.

Under the weighted matrix factorization framework, we extend each tweet by its t nearest neighbors. Specifically, for each tweet, we incorporate additional words from its neighboring tweets. The values of the new words are averaged. Moreover, these new words are treated differently by assigning a new weight w_n to them, since we believe that the new words are not as informative as the original words in the tweet.

We present an illustrative example of how to use neighbors to extend the tweets. Let $\boldsymbol{x}_1$ be a tweet with the following words (the numbers after the colon are TF-IDF values):

$\boldsymbol{x}_1$ = {obama:5.5, medicare:8.3, website:3.8}

which has two nearest neighbors:

$\boldsymbol{x}_{27}$ = {obama:5.5, medicare:8.3, website:3.8, down:5.4}

$\boldsymbol{x}_{356}$ = {obama:5.5, medicare:8.3, website:3.8, problem:7.0}

Then there are two additional words added in $\boldsymbol{x}_1$ whose values are averaged. The new data vector $\boldsymbol{x}'_1$ is:

$\boldsymbol{x}'_1$ = {obama:5.5, medicare:8.3, website:3.8, down:2.7, problem:3.5}

Therefore, the algorithm is run on the new neighbor-augmented data matrix, denoted by X', and the weight matrix W becomes

$$W_{i,j} = \begin{cases} 1, & \text{if } X'_{ij} \neq 0 \ \&j \text{ is an original word,} \\ w_n, & \text{if } X'_{ij} \neq 0, \ \&j \text{ is from neighbor tweets,} \\ w_m, & \text{if } X'_{ij} = 0. \end{cases} \tag{6}$$

This model is referred to as Orthogonal Matrix Factorization with Neighbors (OrMFN).

5.2 Binary coding without Neighbors

It is important to point out that the data used by OrMFN, X', could be a very small subset of the whole dataset. Therefore we only need to find neighbors for a small portion of the data. After the P matrix is learned, the neighborhood information is implicitly encoded in the matrix P, and we still apply the same binarization function $\text{sgn}({P_{\cdot,k}}^\top \bar{\boldsymbol{x}})$ on the whole dataset (in large scale) **without** neighborhood information. We randomly sample 200,000 tweets for OrMFN to learn P; neighbors are extracted only for these 200,000 tweets (note that the neighbors are from the 200,000 tweets as well), and then we use the learned P to generate binary codes for the whole dataset 1.35 million tweets **without** searching for their nearest neighbors.[3]

Our scheme has a clear advantage: the binary coding remains very efficient. During binarization for any data, there is no need to compare 10,000 most recent tweets to find nearest neighbors, which could be time-consuming. An opposite example is the method presented in (Guo et al., 2013), where t most nearest neighbor tweets were extracted, and a tweet profile $Q_{j,\cdot}$ was explicitly forced to be similar to its neighbors' profiles. However, for each new data, the approach proposed in (Guo et al., 2013) requires computing its nearest neighbors.

6 Experiments

6.1 Tweet Data

We crawled English tweets spanning three months from October 5th 2013 to January 5th 2014 using the Twitter API.[4] We cleaned the data such that each hashtag appears at least 100 times in the corpus, and

[3] When generating the binary codes for the 200,000 tweets, these tweets are not augmented with neighbor words.

[4] https://dev.twitter.com

each word appears at least 10 times. This data collection consists of 1,350,159 tweets, 15 million word tokens, 30,608 unique words, and 3,214 unique hashtags.

One of main reasons to use hashtags is to enhance accessing topically similar tweets (Efron, 2010). In a large-scale data setting, it is impossible to manually identify relevant tweets. Therefore, we use Twitter hashtags to create groundtruth labels, which means that tweets marked by the same hashtag as the query tweet are considered relevant. Accordingly, in our experiments all hashtags are removed from the original data corpus. We chose a subset of hashtags from the most frequent hashtags to create groundtruth labels: we manually removed some tags from the subset that are not topic-related (e.g., *#truth*, *#lol*) or are ambiguous; we also removed all the tags that are referring to TV series (the relevant tweets can be trivially obtained by named entity matching). The resulting subset contains 18 hashtags.[5]

100 tweets are randomly selected as queries (test data) for each of the 18 hashtags. The median number of relevant tweets per query is 5,621. The small size of gold standard makes the task relatively challenging. We need to identify 5,621 (0.42% of the whole dataset) tweets out of 1.35 million tweets.

200,000 tweets are randomly selected (not including the 1,800 queries) as training data for the data dependent models to learn binarization functions.[6] The functions are subsequently applied on all the 1.35 million tweets, including the 1,800 query tweets.

6.2 Evaluation

We evaluate a model by the search quality: given a tweet as query, we would like to rank the relevant tweets as high as possible. Following previous work (Weiss et al., 2008; Liu et al., 2011), we use mean precision among top 1000 returned list (MP@1000) to measure the ranking quality. Let pre@k be the precision among top k return data, then MP@1000 is the average value of pre@1, pre@2...pre@1000. Obviously MP gives more reward on the systems that can rank relevant data in the top places, e.g., if the highest ranked tweet is a relevant tweet, then all the precision values (pre@2, pre@3, pre@4...) are increased. We also calculate the precision and recall curve at varying values of top k returned list.

6.3 Methods

We evaluate the proposed unsupervised binary coding models OrMF and OrMFN, whose performance is compared against 5 other unsupervised methods, LSH, SH, LSA, ITQ, and WTMF. All the binary coding functions except LSH are learned on the 200,000 tweet set. All the methods have the same form of binary coding functions: $\text{sgn}({P_{\cdot,k}}^\top \bar{\boldsymbol{x}})$, where they differ only in the projection vector $P_{\cdot,k}$. The retrieved tweets are ranked according to their Hamming distance to the query, where Hamming distance is the number of different bit positions between the binary codes of a tweet and the query.

For ITQ and SH, we use the code provided by the authors. Note that the dense matrix $\bar{X}\bar{X}^\top$ is impossible to compute due the large vocabulary, therefore we replace it by sparse matrix $XX^\top$. For the three matrix factorization based methods (WTMF, OrMF, OrMFN) we run 10 iterations. The regularizer λ in equation 6 is fixed at 20 as in (Guo and Diab, 2012). A small set of 500 tweets is selected from the training set as tuning set to choose the missing word weight w_m in the baseline WTMF, and then its value is fixed for OrMF and OrMFN. The same 500 tweets tuning set is used to choose the neighbor word weight w_n. In fact these models are very stable, consistently outperforming the baselines regardless of different values of w_m and w_n, as later shown in Figure 4 and 5.

We also present the results of cosine similarity on the original word space (COSINE) as an upper bound of the binary coding methods. We implemented an efficient algorithm for COSINE, which is the algorithm 1 in (Petrovic et al., 2010). It firstly normalizes each data to a unit vector, then cosine similarity is calculated by traversing only once the tweets via inverted word index.

6.4 Results

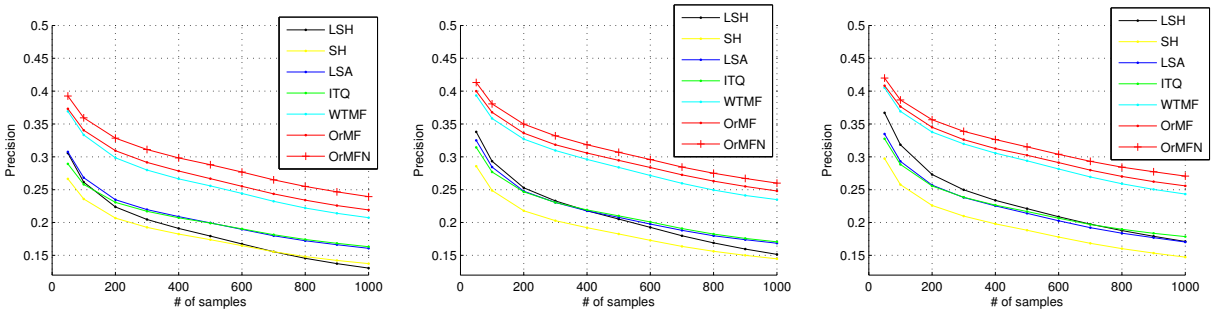
Table 2 summarizes the ranking performance measured by MP@1000 (the mean precision at top 1000 returned list). Figures 2 and 3 illustrate the corresponding precision and recall curve for the Hamming

[5]The tweet dataset and their associated list of hashtags will be available upon request.

[6]Although we use the word "training", the hashtags are never seen by the models. The training data is used for the models to learn the word co-occurrence, and construct binary coding functions.

Models	Parameters	r=64	r=96	r=128
LSH	–	19.21%	21.84%	23.75%
SH	–	18.29%	19.32%	19.95%
LSA	–	21.04%	22.07%	22.67%
ITQ	–	20.8%	22.06%	22.86%
WTMF	$w_m = 0.1$	26.64%	29.39%	30.38%
OrMF	$w_m = 0.1$	27.7%	30.48%	31.26%
OrMFN	$w_m = 0.1, w_n = 0.5$	**29.73%**	**31.73%**	**32.55%**
COSINE	–	33.68%		

Table 2: Mean precision among top 1000 returned list



(a) $r = 64$ (b) $r = 96$ (c) $r = 128$

Figure 2: Hamming ranking: precision curve under top 1000 returned list

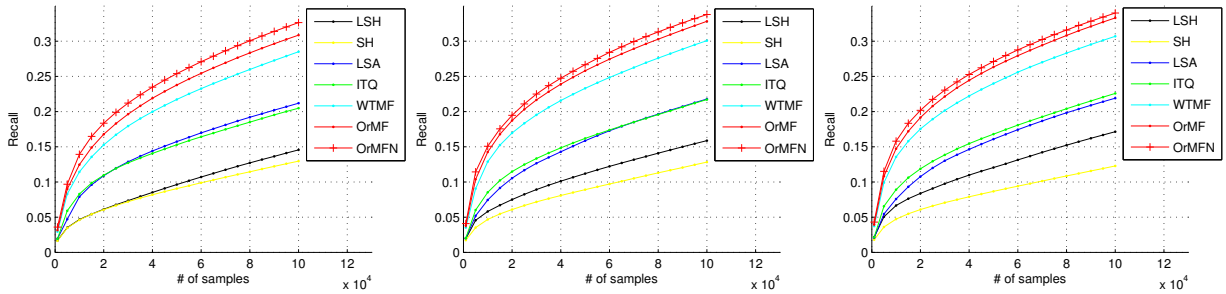
distance ranking. The number of r binary coding functions corresponds to the number of dimensions in the 6 data-dependent models LSA, SH, ITQ, WTMF, OrMF and OrMFN. The missing words weight w_m is fixed as 0.1 based on the tuning set in the three weighted matrix factorization based models WTMF, OrMF and OrMFN. The neighbor word weight w_n is chosen as 0.5 for OrMFN. Later in Section 6.4.1 we show that the performance is robust using varying values of w_m and w_n.

As the number of bits increases, all binary coding models yield better results. This is understandable since the binary bits really record very tiny bits of information from each tweet, and more bits, the more they are able to capture more semantic information.

SH has the worst MP@1000 performance. The reason might be it is designed for vision data where the data vector is relatively dense. ITQ yields comparable results to LSA in terms of MP@1000, yet the recall curve in Figure 3b,c clearly shows the superiority of ITQ over LSA.

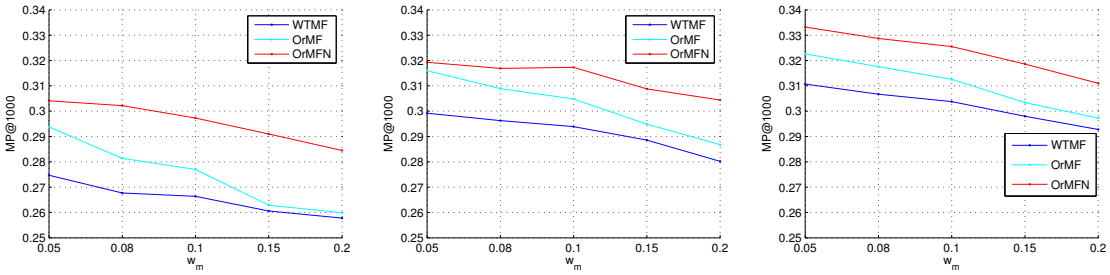
WTMF outperforms LSA by a large margin (around 5% to 7%) through properly modeling missing words, which is also observed in (Guo and Diab, 2012). Although WTMF already reaches a very high MP@1000 performance level, OrMF can still achieve around 1% improvement over WTMF, which can be attributed to orthogonal projections that captures more distinct topics. At last, leveraging neighborhood information, OrMFN is the best performing model (around 1% improvement over OrMF). The trend holds consistently across all conditions. The precision and recall curves in Figures 2 and 3 confirm the trend observed in Table 2 as well.

All the binary coding models yield worse performance than COSINE baseline. This is expected, as the binary bits are employed to gain efficiency at the cost of accuracy: the 128 bits significantly compress the data losing a lot of nuanced information, whereas in the high dimensional word space 128 bits can be only used to record two words (32 bits for two word indices and 32 bits for two TF-IDF values). We manually examined the ranking list. We found in the binary coding models, there exist a lot of ties (128 bits only result in 128 possible Hamming distance values), whereas the COSINE baseline can correctly rank them by detecting the subtle difference signaled by the real-valued TF-IDF values.



(a) $r = 64$ (b) $r = 96$ (c) $r = 128$

Figure 3: Hamming ranking: recall curve under top 100,000 returned list



(a) $r = 64$ (b) $r = 96$ (c) $r = 128$

Figure 4: Weighted matrix factorization based models: MP@1000 vs. missing word weight w_m

6.4.1 Analysis

We are interested in whether other values of w_m and w_n can generate good results – in other words, whether the performance is robust to the two parameter values. Accordingly, we present their impact on MP@1000 in Figure 4 and 5. In Figure 4, the missing word weight w_m is chosen from $\{0.05, 0.08, 0.1, 0.15, 0.2\}$, where in OrMFN the neighbor weight w_n is fixed as 0.5. The figure indicates we can achieve even better MP@1000 around 33.2% when selecting the optimal $w_m = 0.05$. In general, the curves for all the code length are very smooth; the chosen value of w_m does not have a negative impact, e.g., the gain from OrMF over WTMF is always positive.

Figure 5 demonstrates the impact of varying the values of neighbor word weight w_n from $\{0, 0.25, 0.5, 0.75, 1\}$ on OrMFN tested in different r conditions. Note that when $w_n = 0$ indicating that no neighbor information is exploited, the OrMFN model is simply reduced to the OrMF model. Based on the Figure illustration we can conclude that integrating neighboring word information always yields a positive effect, since any value of $w_n > 0$ yields a performance gain over $w_n = 0$ which is OrMF.

6.5 Computation Cost

The data-dependent models involve 2 steps: 1) learning coding functions from a small dataset, and 2) binary coding for the large scale whole dataset.[7] In real-time scenarios, the time is only spent on the 2nd step that involves no matrix factorization. The computation cost of binary coding for all models (LSH, ITQ, LSA, WTMF, OrMF and OrMFN) are roughly the same: $\text{sgn}({P_{\cdot,k}}^\top \bar{\boldsymbol{x}})$. Note that ${P_{\cdot,k}}^\top \bar{\boldsymbol{x}} = {P_{\cdot,k}}^\top \boldsymbol{x} - {P_{\cdot,k}}^\top \mu$ where $\boldsymbol{x}$ is a very sparse vector (with 11 non-zeros values on average) and ${P_{\cdot,k}}^\top \mu$ can be precomputed. On the other hand, calculating Hamming distance on binary codes is also very fast using the logic operations.

[7] Learning the binarization functions can be always done on a small dataset, for example in this paper all the data dependent models are run on the 200,000 tweets, hence it performs very fast. In addition, in the OrMFN model, there is no need to find nearest neighbors for the whole dataset in the 2nd step (the binary coding step).

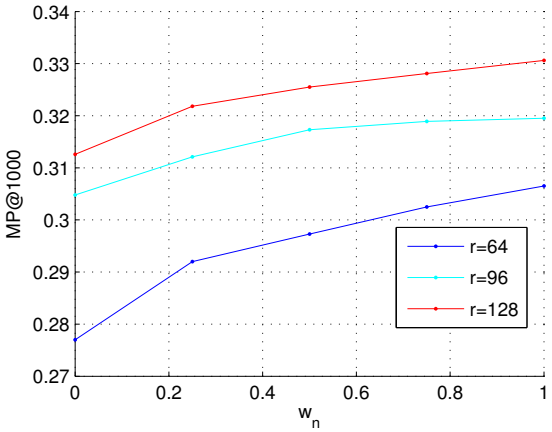


Figure 5: OrMFN model: MP@1000 vs. neighbor word weight w_n

7 Conclusion

In this paper, we proposed a novel unsupervised binary coding model which provides efficient similarity search in massive tweet data. The proposed model, OrMFN, improves an existing matrix factorization model through learning nearly orthogonal projection directions and leveraging the neighborhood information hidden in tweet data. We collected a dataset whose groundtruth labels are created from Twitter hashtags. Our experiments conducted on this dataset showed significant performance gains of OrMFN over the competing methods.

Acknowledgements

We thank Boyi Xie and three anonymous reviewers for their valuable comments. This project is supported by the DARPA DEFT Program.

References

Puneet Agarwal, Rajgopal Vaithiyanathan, Saurabh Sharma, and Gautam Shroff. 2012. Catching the long-tail: Extracting local news events from twitter. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *First Joint Conference on Lexical and Computational Semantics (*SEM)*.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Rahul Bhagat and Deepak Ravichandran. 2008. Large scale acquisition of paraphrases for learning surface patterns. In *Proceedings of ACL-08: HLT*.

Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. 1998. Min-wise independent permutations. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*.

Deepayan Chakrabarti and Kunal Punera. 2011. Event summarization using tweets. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*.

Miles Efron. 2010. Information search and retrieval in microblogs. In *Journal of the American Society for Information Science and Technology*.

Yunchao Gong and Svetlana Lazebnik. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Weiwei Guo, Hao Li, Heng Ji, and Mona Diab. 2013. Linking tweets to news: A framework to enrich online short text data in social media. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*.

Brian Kulis and Kristen Grauman. 2012. Kernelized locality-sensitive hashing. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 34(6):1092–1104.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. In *Psychological review*.

Yuhua Li, David McLean, Zuhair A. Bandar, James D. O'Shea, and Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transaction on Knowledge and Data Engineering*, 18.

Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2011. Hashing with graphs. In *Proceedings of the 28th International Conference on Machine Learning*.

Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012a. Supervised hashing with kernels. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

Wei Liu, Jun Wang, Yadong Mu, Sanjiv Kumar, and Shih-Fu Chang. 2012b. Compact hyperplane hashing with bilinear functions. In *Proceedings of the 29th International Conference on Machine Learning*.

Mohammad Norouzi and David J. Fleet. 2011. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on Machine Learning*.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Nathan Srebro and Tommi Jaakkola. 2003. Weighted low-rank approximations. In *Proceedings of the Twentieth International Conference on Machine Learning*.

Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Julien Subercaze, Christophe Gravier, and Frederique Laforest. 2013. Towards an expressive and scalable twitter's users profiles. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*.

Yair Weiss, Antonio Torralba, and Rob Fergus. 2008. Spectral hashing. In *Advances in Neural Information Processing Systems*.