Two-Stage Hashing for Fast Document Retrieval

Hao Li* **Wei Liu†** **Heng Ji***
*Computer Science Department,
Rensselaer Polytechnic Institute, Troy, NY, USA
{lih13,jih}@rpi.edu
†IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
weiliu@us.ibm.com

Abstract

This work fulfills sublinear time Nearest Neighbor Search (NNS) in massive-scale document collections. The primary contribution is to propose a two-stage unsupervised hashing framework which harmoniously integrates two state-of-the-art hashing algorithms Locality Sensitive Hashing (LSH) and Iterative Quantization (ITQ). LSH accounts for neighbor candidate pruning, while ITQ provides an efficient and effective reranking over the neighbor pool captured by LSH. Furthermore, the proposed hashing framework capitalizes on both term and topic similarity among documents, leading to precise document retrieval. The experimental results convincingly show that our hashing based document retrieval approach well approximates the conventional Information Retrieval (IR) method in terms of retrieving semantically similar documents, and meanwhile achieves a speedup of over one order of magnitude in query time.

1 Introduction

A *Nearest Neighbor Search* (NNS) task aims at searching for top K objects (*e.g.*, documents) which are most similar, based on pre-defined similarity metrics, to a given query object in an existing dataset. NNS is essential in dealing with many search related tasks, and also fundamental to a broad range of Natural Language Processing (NLP) down-stream problems including person name spelling correction (Udupa and Kumar, 2010), document translation pair acquisition (Krstovski and Smith, 2011), large-scale similar noun list generation (Ravichandran et al., 2005), lexical variants mining (Gouws et al., 2011), and large-scale first story detection (Petrovic et al., 2010).

Hashing has recently emerged to be a popular solution to tackling fast NNS, and been successfully applied to a variety of non-NLP problems such as visual object detection (Dean et al., 2013) and recognition (Torralba et al., 2008a; Torralba et al., 2008b), large-scale image retrieval (Kulis and Grauman, 2012; Liu et al., 2012; Gong et al., 2013), and large-scale machine learning (Weiss et al., 2008; Liu et al., 2011; Liu, 2012). However, hashing has received limited attention in the NLP field to the date. The basic idea of hashing is to represent each data object as a binary code (each bit of a code is one digit of "0" or "1"). When applying hashing to handle NLP problems, the advantages are two-fold: 1) the capability to store a large quantity of documents in the main memory. for example, one can store 250 million documents with 1.9G memory using only 64 bits for each document while a large news corpus such as the English Gigaword fifth edition[1] stores 10 million documents in a 26G hard drive; 2) the time efficiency of manipulating binary codes, for example, computing the hamming distance between a pair of binary codes is several orders of magnitude faster than computing the real-valued cosine similarity over a pair of document vectors.

The early explorations of hashing focused on using random permutations or projections to construct randomized hash functions, *e.g.*, the well-known Min-wise Hashing (MinHash) (Broder et al., 1998) and Locality Sensitive Hashing (LSH) (Andoni and Indyk, 2008). In contrast to such data-independent hashing schemes, recent research has been geared to studying data-dependent hashing through learning compact hash codes from a training dataset. The state-of-the-art unsupervised learning-based hashing methods include Spectral Hashing (SH) (Weiss et al., 2008), Anchor Graph Hashing (AGH) (Liu et al., 2011), and Iterative Quantization (ITQ) (Gong et al.,

[1] http://catalog.ldc.upenn.edu/LDC2011T07

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 495–500,
Baltimore, Maryland, USA, June 23-25 2014.

2013), all of which endeavor to make the learned hash codes preserve or reveal some intrinsic structure, such as local neighborhood structure, low-dimensional manifolds, or the closest hypercube, underlying the training data. Despite achieving data-dependent hash codes, most of these "learning to hash" methods cannot guarantee a high success rate of looking a query code up in a hash table (referred to as hash table lookup in literature), which is critical to the high efficacy of exploiting hashing in practical uses. It is worth noting that we choose to use ITQ in the proposed two-stage hashing framework for its simplicity and efficiency. ITQ has been found to work better than SH by Gong et al. (2013), and be more efficient than AGH in terms of training time by Liu (2012).

To this end, in this paper we propose a novel two-stage unsupervised hashing framework to simultaneously enhance the hash lookup success rate and increase the search accuracy by integrating the advantages of both LSH and ITQ. Furthermore, we make the hashing framework applicable to combine different similarity measures in NNS.

2 Background and Terminology

- **Binary Codes:** A bit (a single bit is "0" or "1") sequence assigned to represent a data object. For example, represent a document as a 8-bit code "11101010".
- **Hash Table:** A linear table in which all binary codes of a data set are arranged to be table indexes, and each table bucket contains the IDs of the data items sharing the same code.
- **Hamming Distance:** The number of bit positions in which bits of the two codes differ.
- **Hash Table Lookup:** Given a query q with its binary code $\boldsymbol{h}_q$, find the candidate neighbors in a hash table such that the Hamming distances from their codes to $\boldsymbol{h}_q$ are no more than a small distance threshold ϵ. In practice ϵ is usually set to 2 to maintain the efficiency of table lookups.
- **Hash Table Lookup Success Rate:** Given a query q with its binary code $\boldsymbol{h}_q$, the probability to find at least one neighbor in the table buckets whose corresponding codes (*i.e.*, indexes) are within a Hamming ball of radius ϵ centered at $\boldsymbol{h}_q$.
- **Hamming Ranking:** Given a query q with its binary code $\boldsymbol{h}_q$, rank all data items according to the Hamming distances between their codes and $\boldsymbol{h}_q$; the smaller the Hamming distance, the higher the data item is ranked.

3 Document Retrieval with Hashing

In this section, we first provide an overview of applying hashing techniques to a document retrieval task, and then introduce two unsupervised hashing algorithms: LSH acts as a neighbor-candidate filter, while ITQ works towards precise reranking over the candidate pool returned by LSH.

3.1 Document Retrieval

The most traditional way of retrieving nearest neighbors for documents is to represent each document as a term vector of which each element is the *tf-idf* weight of a term. Given a query document vector $\boldsymbol{q}$, we use the *Cosine* similarity measure to evaluate the similarity between $\boldsymbol{q}$ and a document $\boldsymbol{x}$ in a dataset:

$$sim(\boldsymbol{q}, \boldsymbol{x}) = \frac{\boldsymbol{q}^\top \boldsymbol{x}}{\|\boldsymbol{q}\|\|\boldsymbol{x}\|}. \quad (1)$$

Then the traditional document retrieval method exhaustively scans all documents in the dataset and returns the most similar ones. However, such a brute-force search does not scale to massive datasets since the search time complexity for each query is $O(n)$; additionally, the computational cost spent on Cosine similarity calculation is also nontrivial.

3.2 Locality Sensitive Hashing

The core idea of LSH is that if two data points are close, then after a "projection" operation they will remain close. In other words, similar data points are more likely to be mapped into the same bucket with a high collision probability. In a typical LSH setting of k bits and L hash tables, a query point $\boldsymbol{q} \in \mathbb{R}^d$ and a dataset point $\boldsymbol{x} \in \mathbb{R}^d$ collide if and only if

$$h_{ij}(\boldsymbol{q}) \equiv h_{ij}(\boldsymbol{x}), i \in [1:L], j \in [1:k], \quad (2)$$

where the hash function $h_{ij}(\cdot)$ is defined as

$$h_{ij}(\boldsymbol{x}) = \operatorname{sgn}\left(\boldsymbol{w}_{ij}^\top \boldsymbol{x}\right), \quad (3)$$

in which $\boldsymbol{w}_{ij} \in \mathbb{R}^d$ is a random projection direction with components being independently and identically drawn from a normal distribution, and the sign function $\operatorname{sgn}(x)$ returns 1 if $x > 0$ and -1 otherwise. Note that we use "1/-1" bits for derivations and training, and "1/0" bits for the hashing

implementation including converting data to binary codes, arranging binary codes into hash tables, and hash table lookups.

3.3 Iterative Quantization

The central idea of ITQ is to learn the binary codes achieving the lowest quantization error that encoding raw data to binary codes incurs. This is pursued by seeking a rotation of the zero-centered projected data. Suppose that a set of n data points $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$ are provided. The data matrix is defined as $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n \times d}$. In order to reduce the data dimension from d to the desired code length $c < d$, Principal Component Analysis (PCA) or Latent Semantic Analysis (LSA) is first applied to $\mathbf{X}$. We thus obtain the zero-centered projected data matrix as $\mathbf{V} = (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)\mathbf{X}\mathbf{U}$ where $\mathbf{U} \in \mathbb{R}^{d \times c}$ is the projection matrix.

After the projection operation, ITQ minimizes the quantization error as follows

$$\mathbb{Q}(\mathbf{B}, \mathbf{R}) = \|\mathbf{B} - \mathbf{V}\mathbf{R}\|_{\mathrm{F}}^2, \tag{4}$$

where $\mathbf{B} \in \{1, -1\}^{n \times c}$ is the code matrix each row of which contains a binary code, $\mathbf{R} \in \mathbb{R}^{c \times c}$ is the target orthogonal rotation matrix, and $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. Finding a local minimum of the quantization error in Eq. (4) begins with a random initialization of $\mathbf{R}$, and then employs a K-means clustering like iterative procedure. In each iteration, each (projected) data point is assigned to the nearest vertex of the binary hypercube, and $\mathbf{R}$ always satisfying $\mathbf{R}\mathbf{R}^\top = \mathbf{I}$ is subsequently updated to minimize the quantization loss given the current assignment; the two steps run alternatingly until a convergence is encountered. Concretely, the two updating steps are:

1. **Fix R and update B:** minimize the following quantization loss

$$\begin{aligned}\mathbb{Q}(\mathbf{B}, \mathbf{R}) &= \|\mathbf{B}\|_{\mathrm{F}}^2 + \|\mathbf{V}\mathbf{R}\|_{\mathrm{F}}^2 - 2\mathrm{tr}\big(\mathbf{R}^\top \mathbf{V}^\top \mathbf{B}\big) \\ &= nc + \|\mathbf{V}\|_{\mathrm{F}}^2 - 2\mathrm{tr}\big(\mathbf{R}^\top \mathbf{V}^\top \mathbf{B}\big) \\ &= constant - 2\mathrm{tr}\big(\mathbf{R}^\top \mathbf{V}^\top \mathbf{B}\big),\end{aligned} \tag{5}$$

   achieving $\mathbf{B} = \mathrm{sgn}(\mathbf{V}\mathbf{R})$;

2. **Fix B and update R:** perform the SVD of the matrix $\mathbf{V}^\top \mathbf{B} \in \mathbb{R}^{c \times c}$ to obtain $\mathbf{V}^\top \mathbf{B} = \mathbf{S}\Omega\widehat{\mathbf{S}}^\top$, and then set $\mathbf{R} = \mathbf{S}\widehat{\mathbf{S}}^\top$.

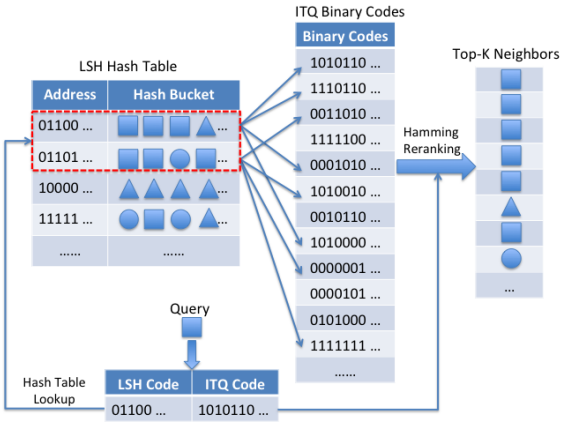

Figure 1: The two-stage hashing framework.

3.4 Two-Stage Hashing

There are three main merits of LSH. (1) It tries to preserve the Cosine similarity of the original data with a probabilistic guarantee (Charikar, 2002). (2) It is training free, and thus very efficient in hashing massive databases to binary codes. (3) It has a very high hash table lookup success rate. For example, in our experiments LSH with more than one hash table is able to achieve a perfect 100% hash lookup success rate. Unfortunately, its drawback is the low search precision that is observed even with long hash bits and multiple hash tables.

ITQ tries to minimize the quantization error of encoding data to binary codes, so its advantage is the high quality (potentially high precision of Hamming ranking) of the produced binary codes. Nevertheless, ITQ frequently suffers from a poor hash lookup success rate when longer bits (*e.g.*, $\geq$ 48) are used (Liu, 2012). For example, in our experiments ITQ using 384 bits has a 18.47% hash lookup success rate within Hamming radius 2. Hence, Hamming ranking (costing $O(n)$ time) must be invoked for the queries for which ITQ fails to return any neighbors via hash table lookup, which makes the searches inefficient especially on very large datasets.

Taking into account the above advantages and disadvantages of LSH and ITQ, we propose a two-stage hashing framework to harmoniously integrate them. Fig. 1 illustrates our two-stage framework with a toy example where identical shapes denote ground-truth nearest neighbors.

In this framework, LSH accounts for neighbor candidate pruning, while ITQ provides an efficient and effective reranking over the neighbor pool captured by LSH. To be specific, the pro-

posed framework enjoys two advantages:

1. Provide a simple solution to accomplish both a high hash lookup success rate and high precision, which does not require scanning the whole list of the ITQ binary codes but scanning the short list returned by LSH hash table lookup. Therefore, a high hash lookup success rate is attained by the LSH stage, while maintaining high search precision due to the ITQ reranking stage.

2. Enable a hybrid hashing scheme combining two similarity measures. The term similarity is used during the LSH stage that directly works on document tf-idf vectors; during the ITQ stage, the topic similarity is used since ITQ works on the topic vectors obtained by applying Latent semantic analysis (LSA) (Deerwester et al., 1990) to those document vectors. LSA (or PCA), the first step in running ITQ, can be easily accelerated via a simple sub-selective sampling strategy which has been proven theoretically and empirically sound by Li et al. (2014). As a result, the nearest neighbors returned by the two-stage hashing framework turns out to be both lexically and topically similar to the query document. To summarize, the proposed two-stage hashing framework works in an unsupervised manner, achieves a sublinear search time complexity due to LSH, and attains high search precision thanks to ITQ. After hashing all data (documents) to LSH and ITQ binary codes, we do not need to save the raw data in memory. Thus, our approach can scale to gigantic datasets with compact storage and fast search speed.

4 Experiments

Data and Evaluations

For the experiments, we use the English portion of the standard TDT-5 dataset, which consists of $278,109$ documents from a time spanning April 2003 to September 2003. 126 topics are annotated with an average of 51 documents per topic, and other unlabeled documents are irrelevant to them. We select six largest topics for the top-K NNS evaluation, with each including more than 250 documents. We randomly select 60 documents from each of the six topics for testing. The six topics are (1). Bombing in Riyadh, Saudi Arabia (2). Mad cow disease in North America (3). Casablanca bombs (4). Swedish Foreign Minister killed (5). Liberian former president arrives in exile and (6). UN official killed in attack. For each document, we apply the Stanford Tokenizer [2] for tokenization; remove stopwords based on the stop list from InQuery (Callan et al., 1992), and apply Porter Stemmer (Porter, 1980) for stemming.

If one retrieved document shares the same topic label with the query document, they are true neighbors. We evaluate the precision of the top-K candidate documents returned by each method and calculate the average precision across all queries.

Results

We first evaluate the quality of term vectors and ITQ binary codes by conducting the whole list Cosine similarity ranking and hamming distance ranking, respectively. For each query document, the top-K candidate documents with highest Cosine similarity scores and shortest hamming distances are returned, then we calculate the average precision for each K. Fig. 2(a) demonstrates that ITQ binary codes could preserve document similarities as traditional term vectors. It is interesting to notice that ITQ binary codes are able to outperform traditional term vectors. It is mainly because some documents are topically related but share few terms thus their relatedness can be captured by LSA. Fig. 2(a) also shows that the NNS precision keep increasing as longer ITQ code length is used and is converged when ITQ code length equals to 384 bits. Therefore we set ITQ code length as 384 bits in the rest of the experiments.

Fig. 2(b) - Fig. 2(e) show that our two-stage hashing framework surpasses LSH with a large margin for both small K (*e.g.,* $K \leq 10$) and large K (*e.g.,* $K \geq 100$) in top-K NNS. It also demonstrates that our hashing based document retrieval approach with only binary codes from LSH and ITQ well approximates the conventional IR method. Another crucial observation is that with ITQ reranking, a small number of LSH hash tables is needed in the pruning step. For example, LSH(40bits) + ITQ(384bits) and LSH(48bits) + ITQ(384bits) are able to reach convergence with only four LSH hash tables. In that case, we can alleviate one main drawback of LSH as it usually requires a large number of hash tables to maintain the hashing quality.

Since the LSH pruning time can be ignored, the search time of the two-stage hashing scheme equals to the time of hamming distance reranking in ITQ codes for all candidates produced from LSH pruning step, *e.g.,* LSH(48bits, 4 tables) +

[2] http://nlp.stanford.edu/software/corenlp.shtml

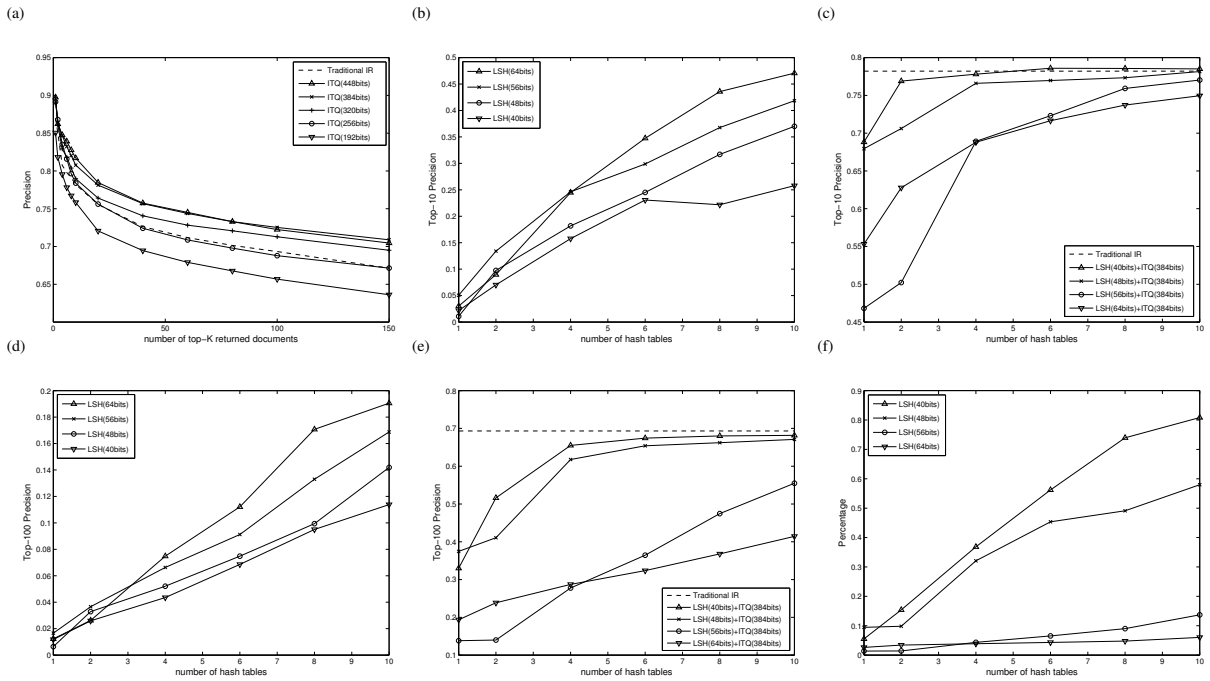

Figure 2: (a) ITQ code quality for different code length, (b) LSH Top-10 Precision, (c) LSH + ITQ(384bits) Top-10 Precision, (d) LSH Top-100 Precision, (e) LSH + ITQ(384bits) Top-100 Precision, (f) The percentage of visited data samples by LSH hash lookup.

ITQ(384bits) takes only one thirtieth of the search time as the traditional IR method. Fig. 2 (f) shows the ITQ data reranking percentage for different LSH bit lengths and table numbers. As the LSH bit length increases or the hash table number decreases, a lower percentage of the candidates will be selected for reranking, and thus costs less search time.

The percentage of visited data samples by LSH hash lookup is a key factor that influence the NNS precision in the two-stage hashing framework. Generally, higher rerank percentage results in better top-K NNS Precision. Further more, by comparing Fig. 2 (c) and (e), it shows that our framework works better for small K than for large K. For example, scanning 5.52% of the data is enough for achieving similar top-10 NNS result as the traditional IR method while 36.86% of the data is needed for top-100 NNS. The reason of the lower performance with large K is that some true neighbors with the same topic label do not share high term similarities and may be filtered out in the LSH step when the rerank percentage is low.

5 Conclusion

In this paper, we proposed a novel two-stage unsupervised hashing framework for efficient and effective nearest neighbor search in massive document collections. The experimental results have shown that this framework achieves not only comparable search accuracy with the traditional IR method in retrieving semantically similar documents, but also an order of magnitude speedup in search time. Moreover, our approach can combine two similarity measures in a hybrid hashing scheme, which is beneficial to comprehensively modeling the document similarity. In our future work, we plan to design better data representation which can well fit into the two-stage hashing theme; we also intend to apply the proposed hashing approach to more informal genres (*e.g.,* tweets) and other down-stream NLP applications (*e.g.,* first story detection).

Acknowledgements

This work was supported by the U.S. ARL No. W911NF-09-2-0053 (NSCTA), NSF IIS-0953149, DARPA No. FA8750- 13-2-0041, IBM, Google and RPI. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

A. Andoni and P. Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122.

A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. 1998. Min-wise independent permutations. In *Proc. STOC*.

J. P. Callan, W. B. Croft, and S. M. Harding. 1992. The inquery retrieval system. In *Proc. the Third International Conference on Database and Expert Systems Applications*.

M. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proc. STOC*.

T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. 2013. Fast, accurate detection of 100,000 object classes on a single machine. In *Proc. CVPR*.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929.

S. Gouws, D. Hovy, and D. Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proc. EMNLP*.

K. Krstovski and D. A. Smith. 2011. A minimally supervised approach for detecting and ranking document translation pairs. In *Proc. the sixth ACL Workshop on Statistical Machine Translation*.

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

Y. Li, C. Chen, W. Liu, and J. Huang. 2014. Sub-selective quantization for large-scale image search. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*.

W. Liu, J. Wang, S. Kumar, and S.-F. Chang. 2011. Hashing with graphs. In *Proc. ICML*.

W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. 2012. Supervised hashing with kernels. In *Proc. CVPR*.

W. Liu. 2012. Large-scale machine learning for classification and search. In *PhD Thesis, Graduate School of Arts and Sciences, Columbia University*.

S. Petrovic, M. Osborne, and V. Lavrenko. 2010. Streaming first story detection with application to twitter. In *Proc. HLT-NAACL*.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

D. Ravichandran, P. Pantel, and E. H. Hovy. 2005. Randomized algorithms and nlp: Using locality sensitive hash functions for high speed noun clustering. In *Proc. ACL*.

A. Torralba, R. Fergus, and W. T. Freeman. 2008a. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970.

A. Torralba, R. Fergus, and Y. Weiss. 2008b. Small codes and large image databases for recognition. In *Proc. CVPR*.

R. Udupa and S. Kumar. 2010. Hashing-based approaches to spelling correction of personal names. In *Proc. EMNLP*.

Y. Weiss, A. Torralba, and R. Fergus. 2008. Spectral hashing. In *NIPS 21*.