Variable Bit Quantisation for LSH

Sean Moran
School of Informatics
The University of Edinburgh
EH8 9AB, Edinburgh, UK
sean.moran@ed.ac.uk

Victor Lavrenko
School of Informatics
The University of Edinburgh
EH8 9AB, Edinburgh, UK
vlavrenk@inf.ed.ac.uk

Miles Osborne
School of Informatics
The University of Edinburgh
EH8 9AB, Edinburgh, UK
miles@inf.ed.ac.uk

Abstract

We introduce a scheme for optimally allocating a variable number of bits per LSH hyperplane. Previous approaches assign a constant number of bits per hyperplane. This neglects the fact that a subset of hyperplanes may be more informative than others. Our method, dubbed Variable Bit Quantisation (VBQ), provides a data-driven non-uniform bit allocation across hyperplanes. Despite only using a fraction of the available hyperplanes, VBQ outperforms uniform quantisation by up to 168% for retrieval across standard text and image datasets.

1 Introduction

The task of retrieving the nearest neighbours to a given query document permeates the field of Natural Language Processing (NLP). Nearest neighbour search has been used for applications as diverse as automatically detecting document translation pairs for the purposes of training a statistical machine translation system (SMT) (Krstovski and Smith, 2011), the large-scale generation of noun similarity lists (Ravichandran et al., 2005) to an unsupervised method for extracting domain specific lexical variants (Stephan Gouws and Metzle, 2011).

There are two broad approaches to nearest neighbour based search: *exact* and *approximate* techniques, which are differentiated by their ability to return completely correct nearest neighbours (the exact approach) or have some possibility of returning points that are not true nearest neighbours (the approximate approach). Approximate nearest neighbour (ANN) search using hashing techniques has recently gained prominence within NLP. The hashing-based approach maps the data into a substantially more compact representation referred to as a *fingerprint*, that is more efficient for performing similarity computations. The resulting compact binary representation radically reduces memory requirements while also permitting fast sub-linear time retrieval of approximate nearest neighbours.

Hashing-based ANN techniques generally comprise two main steps: a *projection* stage followed by a *quantisation* stage. The projection stage performs a neighbourhood preserving embedding, mapping the input data into a lower-dimensional representation. The quantisation stage subsequently reduces the cardinality of this representation by converting the real-valued projections to binary. Quantisation is a lossy transformation which can have a significant impact on the resulting quality of the binary encoding.

Previous work has quantised each projected dimension into a uniform number of bits (Indyk and Motwani, 1998) (Kong and Li, 2012) (Kong et al., 2012) (Moran et al., 2013). We demonstrate that uniform allocation of bits is sub-optimal and propose a data-driven scheme for variable bit allocation. Our approach is distinct from previous work in that it provides a general objective function for bit allocation. VBQ makes no assumptions on the data and, in addition to LSH, it applies to a broad range of other projection functions.

2 Related Work

Locality sensitive hashing (LSH) (Indyk and Motwani, 1998) is an example of an approximate nearest neighbour search technique that has been widely used within the field of NLP to preserve the Cosine distances between documents (Charikar, 2002). LSH for cosine distance draws a large number of random hyperplanes within the input feature space, effectively dividing the space into non-overlapping regions (or buckets). Each hyperplane contributes one bit to the encoding, the value (0 or 1) of which is determined by comput-

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 753–758,
Sofia, Bulgaria, August 4-9 2013.

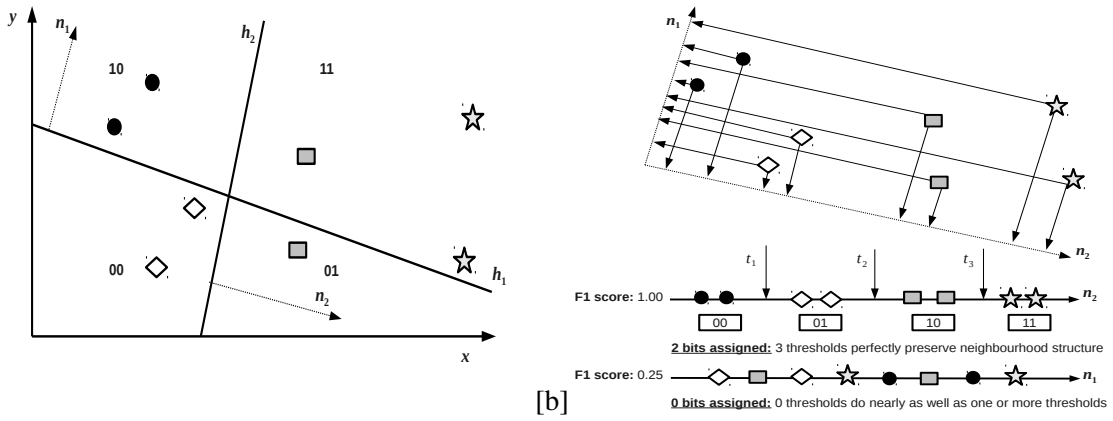

Figure 1: **Left:** Data points with identical shapes are 1-NN. Two hyperplanes $\mathbf{h_1}$, $\mathbf{h_2}$ are shown alongside their associated normal vectors ($\mathbf{n_1}$, $\mathbf{n_2}$). **Right top:** Projection of points onto the normal vectors $\mathbf{n_1}$ and $\mathbf{n_2}$ of the hyperplanes (arrows denote projections). **Right middle:** Positioning of the points along normal vector $\mathbf{n}_2$. Three quantisation thresholds (t_1, t_2, t_3, and consequently 2 bits) can maintain the neighbourhood structure. **Right bottom:** the high degree of mixing between the 1-NN means that this hyperplane ($\mathbf{h_1}$) is likely to have 0 bits assigned (and therefore be discarded entirely).

ing the dot product of a data-point ($\mathbf{x}$) with the normal vector to the hyperplane ($\mathbf{n_i}$): that is, if $\mathbf{x}.\mathbf{n_i} < 0$, $i \in \{1 \dots k\}$, then the i-th bit is set to 0, and 1 otherwise. This encoding scheme is known as single bit quantisation (SBQ). More recent hashing work has sought to inject a degree of data-dependency into the positioning of the hyperplanes, for example, by using the principal directions of the data (Wang et al., 2012) (Weiss et al., 2008) or by training a stack of restricted Boltzmann machines (Salakhutdinov and Hinton, 2009).

Existing quantisation schemes for LSH allocate either one bit per hyperplane (Indyk and Motwani, 1998) or multiple bits per hyperplane (Kong et al., 2012) (Kong and Li, 2012) (Moran et al., 2013). For example, (Kong et al., 2012) recently proposed the Manhattan Hashing (MQ) quantisation technique where each projected dimension is encoded with multiple bits of natural binary code (NBC). The Manhattan distance between the NBC encoded data points is then used for nearest neighbour search. The authors demonstrated that MQ could better preserve the neighbourhood structure between the data points as compared to SBQ with Hamming distance.

Other recent quantisation work has focused on the setting of the quantisation thresholds: for example (Kong and Li, 2012) suggested encoding each dimension into two bits and using an adaptive thresholding scheme to set the threshold positions. Their technique dubbed, Double Bit Quantisation (DBQ), attempts to avoid placing thresholds between data points with similar projected values. In other work (Moran et al., 2013) demonstrated that retrieval accuracy could be enhanced by using a topological quantisation matrix to guide the quantisation threshold placement along the projected dimensions. This topological quantisation matrix specified pairs of ϵ-nearest neighbours in the original feature space. Their approach, Neighbourhood Preserving Quantisation (NPQ), was shown to achieve significant increases in retrieval accuracy over SBQ, MQ and DBQ for the task of image retrieval. In all of these cases the bit allocation is *uniform*: each hyperplane is assigned an identical number of bits.

3 Variable Bit Quantisation

Our proposed quantisation scheme, Variable Bit Quantisation (VBQ), assigns a variable number of bits to each hyperplane subject to a maximum upper limit on the total number of bits[1]. To do so, VBQ computes an F-measure based directly on the positioning of the quantisation thresholds along a projected dimension. The higher the F-measure for a given hyperplane, the better that hyperplane is at preserving the neighbourhood structure between the data points, and the more bits the hyperplane should be afforded from the bit budget B.

Figure 1(a) illustrates the original 2-dimensional feature space for a toy example.

[1]Referred to as the bit budget B, typically 32 or 64 bits.

The space is divided into 4 buckets by two random LSH hyperplanes. The circles, diamonds, squares and stars denote 1-nearest neighbours (1-NN). Quantisation for LSH is performed by projecting the data points onto the normal vectors ($\mathbf{n}_1$, $\mathbf{n}_2$) to the hyperplanes ($\mathbf{h}_1$, $\mathbf{h}_2$). This leads to two *projected dimensions*. Thresholding these projected dimensions at zero, and determining which side of zero a given data-point falls, yields the bit encoding for a given data-point.

Figure 1(b) demonstrates our proposed quantisation scheme. Similar to vanilla LSH, the datapoints are projected onto the normal vectors, to yield two projected dimensions. This is illustrated on the topmost diagram in Figure 1(b). VBQ differs in how these projected dimensions are thresholded to yield the bit encoding: rather than one threshold situated at zero, VBQ employs one or more thresholds and positions these thresholds in an adaptive manner based upon maximisation of an F-measure. Using multiple thresholds enables more than one bit to be assigned per hyperplane[2].

Figure 1(b) (middle, bottom) depicts the F-measure driven threshold optimisation along the projected dimensions. We define as a *positive pair*, those pairs of data points in the original feature space that are ϵ-nearest neighbours (ϵ-NN), and a *negative pair* otherwise. In our toy example, data points with the same shape symbol form a positive pair, while points with different symbols are negative pairs. Intuitively, the thresholds should be positioned in such a way as to maximize the number of positive pairs that fall within the same thresholded region, while also ensuring the negative pairs fall into different regions.

This intuition can be captured by an F-measure which counts the number of positive pairs that are found within the same thresholded regions (true positives, TP), the number of negative pairs found within the same regions (false positives, FP), and the number of positive pairs found in different regions of the threshold partitioned dimension (false negatives, FN). For $\mathbf{n}_2$, three thresholds are optimal, given they perfectly preserve the neighbourhood structure. For $\mathbf{n}_1$, no thresholds can provide a neighbourhood preserving quantisation and therefore it is better to discard the hyperplane $\mathbf{h}_1$. VBQ uses random restarts to optimise the F-measure[3].

The computed F-measure scores per hyperplane (h), per bit count (b) are an effective signal for bit allocation: more informative hyperplanes tend to have higher F-measure, for higher bit counts. VBQ applies a binary integer linear program (BILP) on top of the F-measure scores to obtain the bit allocation. To do so, the algorithm collates the scores in a matrix $\mathbf{F}$ with elements $F_{b,h}$, where $b \in \{0, \ldots, k\}$ [4] indexes the rows, with k being the maximum number of bits allowable for any given hyperplane (set to 4 in this work), and $h \in \{1 \ldots, B\}$ indexes the columns. The BILP uses $\mathbf{F}$ to find the bit allocation that maximises the cumulative F-measure across the B hyperplanes (Equation 1).

$$\begin{aligned} \max \quad & \|\mathbf{F} \circ \mathbf{Z}\| \\ \text{subject to} \quad & \|\mathbf{Z}_h\| = 1 \qquad h \in \{1 \ldots B\} \\ & \|\mathbf{Z} \circ \mathbf{D}\| \leq B \\ & \mathbf{Z} \quad \text{is binary} \end{aligned} \tag{1}$$

$\|.\|$ denotes the Frobenius L_1 norm, $\circ$ the Hadamard product and $\mathbf{D}$ is a constraint matrix, with $D_{b,h} = b$, ensuring that the bit allocation remains within the bit budget B. The BILP is solved using the standard branch and bound optimization algorithm (Land and Doig, 1960). The output from the BILP is an indicator matrix $\mathbf{Z} \in \{0,1\}^{(k+1)\times B}$ whose columns specify the optimal bit allocation for a given hyperplane i.e. $Z_{b,h} = 1$ if the BILP decided to allocate b bits for hyperplane h, and zero otherwise. Example matrices for the toy problem in Figure 1 are given hereunder (in this example, $k = 2$ and $B = 2$).

$$\begin{array}{lcc} \mathbf{F} & h_1 & h_2 \\ b_0 & 0.25 & 0.25 \\ b_1 & 0.35 & 0.50 \\ b_2 & 0.40 & 1.00 \end{array} \qquad \mathbf{D} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{pmatrix} \qquad \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Notice how the indicator matrix $\mathbf{Z}$ specifies an assignment of 0 bits for hyperplane h_1 and 2 bits for hyperplane h_2 as this yields the highest cumulative F-measure across hyperplanes while also meeting the bit budget. VBQ is therefore a principled method to select a discriminative subset of hyperplanes, and simultaneously allocate bits to the remaining hyperplanes, given a fixed overall bit budget B, while maximizing cumulative F-measure.

[2] b bits, requires $2^b - 1$ thresholds.

[3] More details on the computation of the F-measure per hyperplane can be found in (Moran et al., 2013).

[4] For 0 bits, we compute the F-measure without any thresholds along the projected dimension.

Dataset	CIFAR-10					TDT-2				Reuters-21578			
	SBQ	MQ	DBQ	NPQ	VBQ	SBQ	MQ	DBQ	VBQ	SBQ	MQ	DBQ	VBQ
SIKH	0.042	0.063	0.047	0.090	**0.161**	0.034	0.045	0.031	**0.092**	0.102	0.112	0.087	**0.389**
LSH	0.119	0.093	0.066	0.153	**0.207**	0.189	0.097	0.089	**0.229**	0.276	0.201	0.175	**0.538**
BLSI	0.038	0.135	0.111	0.155	**0.231**	0.283	0.210	0.087	**0.396**	0.100	0.030	0.030	**0.156**
SH	0.051	0.135	0.111	0.167	**0.202**	0.146	0.212	0.167	**0.370**	0.033	0.028	0.030	**0.154**
PCAH	0.036	0.137	0.107	0.153	**0.219**	0.281	0.208	0.094	**0.374**	0.095	0.034	0.027	**0.154**

Table 1: Area under the Precision Recall curve (AUPRC) for all five projection methods. Results are for 32 bits (images) and at 128 bits (text). The best overall score for each dataset is shown in bold face.

4 Experiments

4.1 Datasets

Our text datasets are *Reuters-21578* and *TDT-2*. The original Reuters-21578 corpus contains 21578 documents in 135 categories. We use the *ModApte* version and discard those documents with multiple category labels. This leaves 8,293 documents in 65 categories. The corpus contains 18,933 distinct terms. The TDT-2 corpus consists of 11,201 on-topic documents which are classified into 96 semantic categories. We remove those documents appearing in two or more categories and keep only the largest 30 categories. This leaves 9,394 documents in total with 36,771 distinct terms. Both text datasets are TF-IDF and L2 norm weighted. To demonstrate the generality of VBQ we also evaluate on the *CIFAR-10* image dataset (Krizhevsky, 2009), which consists of 60,000 images represented as 512 dimensional Gist descriptors (Oliva and Torralba, 2001). All of the datasets are identical to those that have been used in previous ANN hashing work (Zhang et al., 2010) (Kong and Li, 2012) and are publicly available on the Internet.

4.2 Projection Methods

VBQ is independent of the projection stage and therefore can be used the quantise the projections from a wide range of different projection functions, including LSH. In our evaluation we take a sample of the more popular data-independent (LSH, SIKH) and data-dependent (SH, PCAH, BLSI) projection functions used in recent hashing work:

- **SIKH:** Shift-Invariant Kernel Hashing (SIKH) uses random projections that approximate shift invariant kernels (Raginsky and Lazebnik, 2009). We follow previous work and use a Gaussian kernel with a bandwidth set to the average distance to the 50th nearest neighbour (Kong et al., 2012) (Raginsky and Lazebnik, 2009).
- **LSH:** Locality Sensitive Hashing uses a Gaussian random matrix for projection (Indyk and Motwani, 1998) (Charikar, 2002).
- **BLSI:** Binarised Latent Semantic Indexing (BLSI) forms projections through Singular Value Decomposition (SVD) (Salakhutdinov and Hinton, 2009).
- **SH:** Spectral Hashing (SH) uses the eigenfunctions computed along the principal component directions of the data for projection (Weiss et al., 2008).
- **PCAH:** Principal Component Analysis Hashing (PCAH) employs the eigenvectors corresponding the the largest eigenvalues of the covariance matrix for projection (Wang et al., 2012).

4.3 Baselines

Single Bit Quantisation (SBQ) (Indyk and Motwani, 1998), Manhattan Hashing (MQ) (Kong et al., 2012), Double Bit Quantisation (DBQ) (Kong and Li, 2012) and Neighbourhood Preserving Quantisation (NPQ) (Moran et al., 2013). MQ, DBQ and NPQ all assign 2 bits per hyperplane, while SBQ assigns 1 bit per hyperplane. All methods, including VBQ, are constrained to be within the allocated bit budget B. If a method assigns more bits to one hyperplane, then it either discards, or assigns less bits to other hyperplanes.

4.4 Evaluation Protocol

We adopt the standard Hamming ranking evaluation paradigm (Kong et al., 2012). We randomly select 1000 query data points per run. Our results are averaged over 10 runs, and the average reported. The ϵ-*neighbours* of each query point

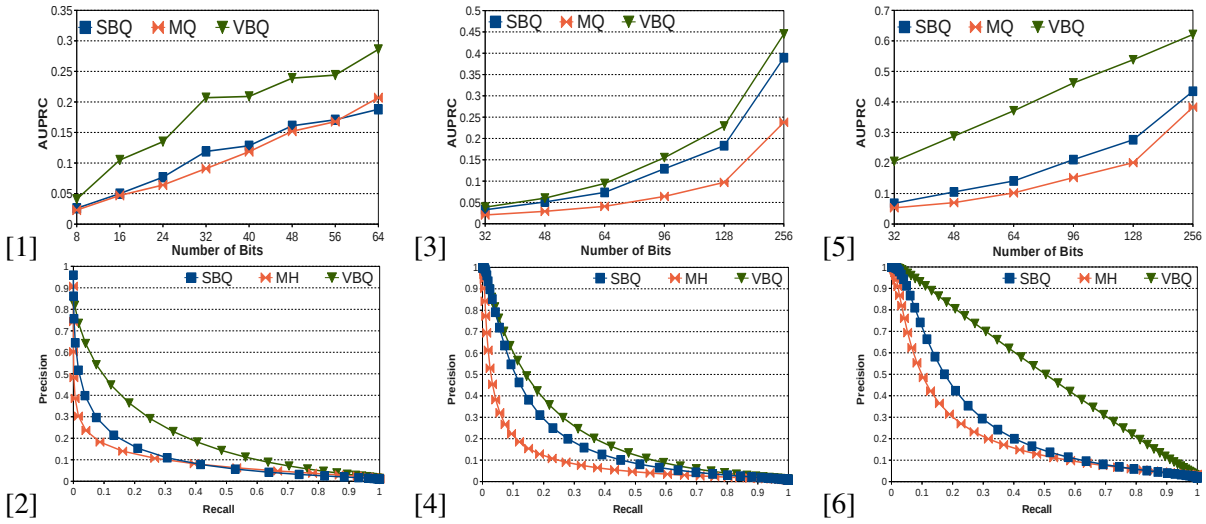

Figure 2: [1] LSH AUPRC vs bits for CIFAR-10 [2] LSH Precision-Recall curve for CIFAR-10 [3] LSH AUPRC vs bits for TDT-2 [4] LSH Precision-Recall curve for TDT-2 [5] LSH AUPRC vs bits for Reuters-21578 [6] LSH Precision-Recall curve for Reuters-21578

form the ground truth for evaluation. The threshold ϵ is computed by sampling 100 training datapoints at random from the training dataset and determining the distance at which these points have 50 nearest neighbours on average. Positive pairs and negative pairs for F-measure computation are computed by thresholding the *training dataset* Euclidean distance matrix by ϵ. We adopt the Manhattan distance and multi-bit binary encoding method as suggested in (Kong et al., 2012). The F-measure we use for threshold optimisation is: $F_\beta = (1+\beta^2)TP/((1+\beta^2)TP+\beta^2 FN+FP)$. We select the parameter β on a *held-out validation* dataset. The area under the precision-recall curve (AUPRC) is used to evaluate the quality of retrieval.

4.5 Results

Table 1 presents our results. For LSH on text (Reuters-21578) at 128 bits we find a substantial 95% gain in retrieval performance over uniformly assigning 1 bit per hyperplane (SBQ) and a 168% gain over uniformly assigning 2 bits per hyperplane (MQ). VBQ gain over SBQ at 128 bits is statistically significant based upon a paired Wilcoxon signed rank test across 10 random train/test partitions (*p-value:* ≤ 0.0054). This pattern is repeated on TDT-2 (for 128 bits, SBQ vs VBQ: *p-value* ≤ 0.0054) and CIFAR-10 (for 32 bits, SBQ vs VBQ: *p-value:* ≤ 0.0054). VBQ also reaps substantial gains for the Eigendecomposition based projections (PCAH, SH, BLSI) effectively exploiting the imbalanced variance across hyperplanes - that is, those hyperplanes capturing higher proportions of the variance in the data are allocated more bits from the fixed bit budget. Figure 2 (top row) illustrates that VBQ is effective across a range of bit budgets. Figure 2 (bottom row) presents the precision-recall (PR) curves at 32 bits (CIFAR-10) and 128 bits (TDT-2, Reuters-21578). We confirm our hypothesis that judicious allocation of variable bits is significantly more effective than uniform allocation.

5 Conclusions

Our proposed quantisation scheme computes a non-uniform bit assignment across LSH hyperplanes. The novelty of our approach is centred upon a binary integer linear program driven by a novel F-measure based objective function that determines the most appropriate bit allocation: hyperplanes that better preserve the neighbourhood structure of the input data points are awarded more bits from a fixed bit budget. Our evaluation on standard datasets demonstrated that VBQ can substantially enhance the retrieval accuracy of a selection of popular hashing techniques across two distinct modalities (text and images). In this paper we concentrated on the hamming ranking based scenario for hashing. In the future, we would like to examine the performance of VBQ in the lookup based hashing scenario where hash tables are used for fast retrieval.

References

Moses Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388.

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*, STOC '98, pages 604–613, New York, NY, USA. ACM.

Weihao Kong and Wu-Jun Li. 2012. Double-bit quantization for hashing. In *AAAI*.

Weihao Kong, Wu-Jun Li, and Minyi Guo. 2012. Manhattan hashing for large-scale image retrieval. SIGIR '12, pages 45–54.

Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. Master's thesis.

Kriste Krstovski and David A. Smith. 2011. A Minimally Supervised Approach for Detecting and Ranking Document Translation Pairs. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, Scotland. Association for Computational Linguistics.

A. H. Land and A. G. Doig. 1960. An automatic method of solving discrete programming problems. *Econometrica*, 28:pp. 497–520.

Sean Moran, Victor Lavrenko, and Miles Osborne. 2013. Neighbourhood preserving quantisation for lsh. In *36th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, Dublin, Ireland, 07/2013.

Aude Oliva and Antonio Torralba. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175.

Maxim Raginsky and Svetlana Lazebnik. 2009. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS '09*, pages 1509–1517.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. ACL '05, pages 622–629. Association for Computational Linguistics.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978.

Dirk Hovy Stephan Gouws and Donald Metzle. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, EMNLP '11, page 8290. Association for Computational Linguistics.

Jun Wang, S. Kumar, and Shih-Fu Chang. 2012. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406.

Yair Weiss, Antonio B. Torralba, and Robert Fergus. 2008. Spectral hashing. In *NIPS*, pages 1753–1760.

Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. 2010. Self-taught hashing for fast similarity search. In *SIGIR*, pages 18–25.