Deep Semantic Text Hashing with Weak Supervision

Suthee Chaidaroon
Department of Computer Engineering
Santa Clara University
Santa Clara, CA, USA
schaidaroon@scu.edu

Travis Ebesu
Department of Computer Engineering
Santa Clara University
Santa Clara, CA, USA
tebesu@scu.edu

Yi Fang
Department of Computer Engineering
Santa Clara University
Santa Clara, CA, USA
yfang@scu.edu

ABSTRACT

With an ever increasing amount of data available on the web, fast similarity search has become the critical component for large-scale information retrieval systems. One solution is semantic hashing which designs binary codes to accelerate similarity search. Recently, deep learning has been successfully applied to the semantic hashing problem and produces high-quality compact binary codes compared to traditional methods. However, most state-of-the-art semantic hashing approaches require large amounts of hand-labeled training data which are often expensive and time consuming to collect. The cost of getting labeled data is the key bottleneck in deploying these hashing methods. Motivated by the recent success in machine learning that makes use of weak supervision, we employ unsupervised ranking methods such as BM25 to extract weak signals from training data. We further introduce two deep generative semantic hashing models to leverage weak signals for text hashing. The experimental results on four public datasets show that our models can generate high-quality binary codes without using hand-labeled training data and significantly outperform the competitive unsupervised semantic hashing baselines.

KEYWORDS

Semantic Hashing, Weak Supervision, Variational Autoencoder

ACM Reference Format:
Suthee Chaidaroon, Travis Ebesu, and Yi Fang. 2018. Deep Semantic Text Hashing with Weak Supervision. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209978.3210090

1 INTRODUCTION

Computing document similarity is one of the most important tasks for large-scale information retrieval problems including web search, document clustering, and recommendations. As massive web information has incurred high computational cost, it is crucial to develop a similarity search solution with computational efficiency and high accuracy. A practical solution is semantic hashing [1] which aims to design compact binary codes to represent documents. It has recently gained much attention since performing similarity search on binary codes only requires an inexpensive Hamming distance function which is counting bitwise differences. Another benefit of using binary codes is memory efficiency as a binary vector can be squeezed into a memory address.

Deep learning has gained tremendous popularity due to its remarkable performance on many important tasks in computer vision, speech recognition, and natural language processing. It can discover novel features from raw input without the need of feature engineering, and deep learning models are often trained efficiently with the backpropagation algorithm on massive high-dimensional data. Recently, deep learning has been successfully applied to the semantic hashing problem for image and text similarity search [2, 3].

Due to the ease of modeling and the excellent performance of supervised hashing models, most deep hashing models are designed to leverage human-labeled data such as tags and categories. When we evaluate the performance of the similarity search using labels as relevant judgments, the supervised learning model can directly map the input documents with the same labels to the similar binary codes. Since the unsupervised hashing models cannot directly observe the ground truth, the models need to exploit the structure of the data to infer the true labels of the documents. In spite of the difficulty in modeling under the unsupervised setting, generating binary codes without hand-labeled data is very practical because acquiring hand-labeled data is costly and time-consuming in many real-world scenarios.

To improve the performance of unsupervised hashing models is quite challenging due to the lack of labels. In this paper, we attempt to utilize the relationships between documents to gather more information from training data. Specifically, we use unsupervised ranking methods such as BM25 [4] to identify the k-nearest documents for every query document. The additional documents serve as a weak signal[1] that may reveal the structure of the document space which can be used to generate a better binary code. However, since the weak signals are often noisy, the learning model must be able to handle the noise introduced by BM25. Due to the success of deep generative models [5] for learning a low-dimensional space from high-dimensional data, we proposed two deep generative models to learn a compact binary code from text documents together with their supplementary neighborhood information. We summarize our main contribution as follows:

- We mitigate the lack of labeled data by using unsupervised ranking to approximate the true document space.
- We design two deep generative models to leverage the contents of the documents and the estimated neighborhood for learning a semantic hash function.

[1] Although some works define a weak supervision as a less precise label, in this paper we refer a weak supervision as using noisy and low-quality data without the need of an external data source.

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210090

- Experiments results on four large public text corpora demonstrate that our models outperforms competitive unsupervised text hashing models.

2 RELATED WORK

There exist extensive studies in learning data-dependent hash functions [2, 6]. Due to the space constraint, we provide a brief review of the two most relevant works to ours. 1) Self-Taught Hashing (STH) [7]: this model aims to preserve the k-nearest neighborhood for each document. The major difference from ours is that this model excludes document features and only considers the document connectivity. 2) The unsupervised Variational Deep Semantic Hashing (VDSH) [3] is a deep generative model that aims to preserve the content of each training document. It has an autoencoder architecture which is similar to ours, but VDSH does not explicitly preserve the document neighborhood.

The use of BM25 as a weak annotator is similar to [8] which uses the output from the BM25 ranking model to train a ranking model based on feedforward neural networks. However, the role of the weak signals is different from ours. We use the weak signals to estimate the true document space while the work [8] uses the weak signals as relevance scores. Another line of works that employs the k-nearest documents as weak signals is the query language modeling. The k-nearest documents are used to solve the vocabulary gap between queries and documents. Some methods such as pseudo-relevance feedback is commonly used and mentioned in the query expansion literature [9].

3 METHODS

3.1 Document Space Estimation

If we have the label information, we can estimate the true document space where all documents with the same labels are nearby. Without the labels, we attempt to use a vector space model to estimate the document space. In the vector space, a document is a fixed-length vector, and the cosine similarity measures the distance between document vectors. Due to the competitive performance of BM25 [4], we represent each document as a bag-of-words vector using the BM25 weighting scheme. For every document $\boldsymbol{d}$ which acts as a pivot document, the BM25 will retrieve the k-nearest documents, denoted as a set of neighbor documents $\text{NN}(\boldsymbol{d})$. We assume that a majority of documents in $\text{NN}(\boldsymbol{d})$ share the same label as its pivot document $\boldsymbol{d}$, hence a binary code of any document within the same proximity in the vector space model should be more similar.

3.2 Neighborhood Recognition Model (NbrReg)

Instead of learning a binary code directly, we will learn a continuous distribution to generate semantic vector $\boldsymbol{s}$ [3]. We use the following generative story for documents:

- Draw a semantic vector $\boldsymbol{s}$ from a standard normal, $\mathcal{N}(\mathbf{0}, \mathbf{1})$.
- For a unique word w_i in a pivot $\boldsymbol{d}$:
  - Draw w_i from $P_{\boldsymbol{A}}(w_i|\mathbf{s})$.
- For each unique word $\tilde{w}_j$ in the neighborhood set $\text{NN}(\boldsymbol{d})$:
  - Draw $\tilde{w}_j$ from $P_{\boldsymbol{B}}(\tilde{w}_j|\mathbf{s})$

$\text{NN}(\boldsymbol{d})$ is a set of the k-nearest documents of the pivot document $\boldsymbol{d}$. We set document likelihood $P(\boldsymbol{d}) = \prod_i P_{\boldsymbol{A}}(w_i|\mathbf{s})$ and neighborhood likelihood $P(\text{NN}(\boldsymbol{d})) = \prod_j P_{\boldsymbol{B}}(\tilde{w}_j|\mathbf{s})$ as a product of word probabilities.

The objective function is obtained by maximizing the variational lowerbound of the log data likelihood of $P(\boldsymbol{d}, \text{NN}(\boldsymbol{d})) = P(\boldsymbol{d})P(\text{NN}(\boldsymbol{d}))$:

$$\begin{aligned}\log P(\boldsymbol{d}, \text{NN}(\boldsymbol{d})) &= \log \int_{\boldsymbol{s}} P(\boldsymbol{d}|\boldsymbol{s})P(\text{NN}(\boldsymbol{d})|\boldsymbol{s})P(\boldsymbol{s})d\boldsymbol{s} \\ &\geq \mathbb{E}_{Q(\boldsymbol{s}|\cdot)}\left[\log P(\boldsymbol{d}|\boldsymbol{s})\right] + \mathbb{E}_{Q(\boldsymbol{s}|\cdot)}\left[\log P(\text{NN}(\boldsymbol{d})|\boldsymbol{s})\right] \\ &\quad - D_{KL}(Q(\boldsymbol{s}|\cdot) \parallel P(\boldsymbol{s})) \end{aligned} \tag{1}$$

where $Q(\boldsymbol{d}|\cdot)$ is an approximate posterior distribution that will be learned from the data. The dot $\cdot$ notation is a placeholder for the input random variables. There are many ways to define $Q(\boldsymbol{d}|\cdot)$; we will define Q in Section 3.4. $P(\boldsymbol{d}|\boldsymbol{s})$ measures how well $\boldsymbol{s}$ represents the global meaning of $\boldsymbol{d}$. Similarly, $P(\text{NN}(\boldsymbol{d})|\boldsymbol{s})$ determines how well $\boldsymbol{s}$ represents the global meaning of the neighborhood of $\boldsymbol{d}$. The Kullback-Leibler divergence (D_{KL}) measures the deviation of $Q(\boldsymbol{d}|\boldsymbol{s})$ from $P(\boldsymbol{s}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$.

3.3 Decoder Function

$P(\boldsymbol{d}|\boldsymbol{s})$ and $P(\text{NN}(\boldsymbol{d})|\boldsymbol{s})$ are viewed as decoder functions and defined as a product of word probabilities[2]:

$$P(\boldsymbol{d}|\boldsymbol{s}) = \prod_i P_{\boldsymbol{A}}(w_i|\boldsymbol{s}) = \prod_i \frac{\exp\{\boldsymbol{s}^T \boldsymbol{A}\boldsymbol{e}_i\}}{\exp\{\sum_j \boldsymbol{s}^T \boldsymbol{A}\boldsymbol{e}_j\}} \tag{2}$$

where $\boldsymbol{e}_i$ is the one-hot encoded vector of the i^{th} unique word w_i in $\boldsymbol{d}$. Matrix $\boldsymbol{A}$ maps semantic vector $\boldsymbol{s}$ to a word embedding space. $P(\text{NN}(\boldsymbol{d})|\boldsymbol{s})$ has a similar definition as Eq.2 while we use a different matrix $\boldsymbol{B}$ as the mapping matrix. The reason to maximize only unique words found in the set of neighbor documents is because we do not want to overcount some bad signals which are words observed in any document that does not have the same label as the pivot document. As a result, we also undercount good words which serve as good signals.

3.4 Encoder Function

We define $Q(\boldsymbol{s}|\cdot)$ as a normal distribution parameterized by the pivot documents $\boldsymbol{d}$: $Q(\boldsymbol{s}|\cdot) = \mathcal{N}(\boldsymbol{s}; f(\boldsymbol{d}))$. Function f maps $\boldsymbol{d}$ to mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$. Since there are two distribution parameters, we need to define mapping function for each parameter: $f = \langle f_\mu, f_\sigma \rangle$. We use feedforward neural networks as a functional form of f_μ and f_σ:

$$f_\mu(\boldsymbol{d}) = \boldsymbol{W}_\mu \cdot h(\boldsymbol{d}) + \boldsymbol{b}_\mu \quad f_\sigma(\boldsymbol{d}) = \boldsymbol{W}_\sigma \cdot h(\boldsymbol{d}) + \boldsymbol{b}_\sigma \tag{3}$$

$$h(\boldsymbol{d}) = \text{relu}\big(\boldsymbol{W}_2 \cdot \text{relu}\big(\boldsymbol{W}_1 \cdot \boldsymbol{d} + \boldsymbol{b}_1\big) + \boldsymbol{b}_2\big) \tag{4}$$

where $\boldsymbol{W}_\cdot$ and $\boldsymbol{b}_\cdot$ are weight and bias of a single-layer feedforward neural network. f_μ and f_σ transform $\boldsymbol{d}$ to $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ so that semantic vector $\boldsymbol{s}$ can be sampled from Q:

[2] We omit the definition of neighborhood likelihood function due to the space limitation.

$$\boldsymbol{s} \sim Q(\boldsymbol{s}|\boldsymbol{d}) = \mathcal{N}(\boldsymbol{s};\ \ \mu = f_\mu(\boldsymbol{d}), \sigma = f_\sigma(\boldsymbol{d})) \quad (5)$$

Computing the gradient of Eq.1 requires estimation. We use Monte Carlo to estimate $\mathbb{E}_{Q(\boldsymbol{s}|\boldsymbol{d})}\left[\log P(\boldsymbol{d}|\boldsymbol{s})\right] \approx \frac{1}{|S|}\sum_{i=1}^{|S|} \log P(\boldsymbol{d}|\boldsymbol{s}^{(i)})$ where $\boldsymbol{s}^{(i)} \sim Q(\boldsymbol{s}|\boldsymbol{d})$. But the gradient estimated by Monte Carlo has a high variance and will be difficult for the models to learn. [5] uses the reparameterization trick to turn distribution $Q(\boldsymbol{s}|\boldsymbol{d})$ to a low-variance gradient estimator[3]:

$$\boldsymbol{s} \sim g(f_\mu(\boldsymbol{d}), f_\sigma(\boldsymbol{d}), \boldsymbol{\epsilon}) = \boldsymbol{\epsilon} \cdot f_\sigma(\boldsymbol{d}) + f_\mu(\boldsymbol{d}),\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \quad (6)$$

Similar to Variational Autoencoder [5], this step introduces another random variable $\boldsymbol{\epsilon}$ as a source of randomness and applies scale and shift on $\boldsymbol{\epsilon}$ to derive $\boldsymbol{s}$ which resulted in a deterministic objective function.

3.5 Utilize Neighbor Documents (NbrReg+Doc)

The NBrReg model generates a semantic vector from the distribution that depends only on the pivot document. However, the neighbor documents also provide useful signals to enhance representation learning. For instance, a set of words used by neighbor documents could indicate a theme of all documents in that region. However, the additional words from neighbor documents are noisy and may confuse the models. To alleviate the influence of bad signals, we use the feedforward neural networks to learn a hidden vector for every document in the neighbor document set and apply a mean pooling layer to compute the centroid of the neighborhood. We modified Eq.3[4] to accommodate the neighbor documents:

$$\boldsymbol{Z}^{\text{NN}} = \text{relu}\big(\boldsymbol{W}_2^{\text{NN}} \cdot \text{relu}\big(\boldsymbol{W}_1^{\text{NN}} \cdot \text{NN}(\boldsymbol{d}) + \boldsymbol{b}_1^{\text{NN}}\big) + \boldsymbol{b}_2^{\text{NN}}\big) \quad (7)$$

$$h_{\text{NN}}(\text{NN}(\boldsymbol{d})) = \text{mean}(\boldsymbol{Z}^{\text{NN}}) \quad (8)$$

$$f_\mu(\boldsymbol{d}, \text{NN}(\boldsymbol{d})) = \boldsymbol{W}_\mu \cdot (h(\boldsymbol{d}) + h_{\text{NN}}(\text{NN}(\boldsymbol{d})) + \boldsymbol{b}_\mu \quad (9)$$

Eq.9 is an encoder function that adds a hidden vector of the pivot document (From Eq.4) with a centroid of its neighborhood. The decoder remains the same as NbrReg model. Consequently, the model takes both pivot and neighbor documents as well as reconstructs the input back. The objective function is the same as Eq.1.

3.6 Binarization

We use the encoder function $Q(\boldsymbol{s}|\cdot)$ to generate a continuous semantic vector for a new document $\boldsymbol{d}$. We take the mean of the encoder function as the output semantic vector, $\bar{\boldsymbol{s}} = \mathbb{E}\left[Q(\boldsymbol{s}|\cdot)\right]$. We use the median method [10] to generate binary code $\boldsymbol{b}$ from $\bar{\boldsymbol{s}}$. That is we set the k^{th} bit of $\boldsymbol{b}$ to 1 if the k^{th} dimension of $\bar{\boldsymbol{s}}$ is larger than the threshold; otherwise, we set the k^{th} bit to 0. We determine the threshold value by computing the median value of the semantic vector in the training set.

4 EXPERIMENTAL SETUP

We use four public text collections for evaluations. 1) **Yahoo! Answers**[5] We select the ten largest categories and use question title, content, and the best answer. The final dataset has 207,261 documents . 2) **AG's news**[6] We select the four largest categories and use the title and description. The resulting data set contains 132,912 news articles. 3) **DBPedia**[7] We select 14 largest classes and sample 107,559 Wikipedia articles. 4) **20Newsgroups**[8] A text collection for text classification comprised of 18K forum posts on 20 topics. For each dataset we split into three subsets with 80% for training, 10% for validation, and 10% for testing.

We compare the performance of the proposed models with the following unsupervised semantic hashing models: VDSH[9] [3], STH[10] [7], Laplacian co-hashing (LCH) [11], Spectral Hashing (SpH) [10], and Locally Sensitive Hashing (LSH)[11] [12]. We used cross-validation to select the hyperparameters for the baselines. The size of the neighborhood for SpH, STH, LCH, and our models is set to 20. Since our goal is to demonstrate the effectiveness of the proposed models without using supervisory signals, we do not include supervised text hashing models. We also include the BM25 model as the additional baseline.

We train the models using training samples and use the encoder function to generate binary codes. We use the Hamming distance to retrieve 100 closest training documents for each test query. Similar to prior works [3, 7, 13, 14], a training document that shares the same label with the test document is considered relevant. We employ average Precision at 100 (Prec@100) which is the ratio of relevant documents over the number of retrieved documents. We report the average Prec@100 over all test documents.

5 EXPERIMENTAL RESULTS

Table 1 presents the performance of all models from 8 to 128 bits. On the large dataset (DBPedia, Yahoo, AgNews), the NbrReg model outperforms the baselines while the NbrReg+Doc model yields the best results on the small dataset (20NG). These results demonstrate that the weak signals generated by BM25 are helpful in learning binary codes.

We observe that NbrReg excels in the large datasets, which indicates that as long as we have enough data encouraging a semantic vector to recognize its neighborhood is more effective than using the neighborhood as the input. Interestingly, the NbrReg+Doc model performs extremely well on 20NG. We believe that when the training set is small, NbrReg+Doc uses more data than NbrReg by taking neighbor documents as an additional input. This allows NbrReg+Doc learns a better representation.

The performance is correlated with the number of bits. As the number of bits increases, the models can pack more relevant information into a semantic vector. When the number of bits is 8 or 16 bits, our models generally perform better than the models that do not use the document's contents such as STH and LCH. This shows that utilizing the contents is useful under a very low-dimensional constraint. Our models also outperform VDSH model

[3] https://ermongroup.github.io/cs228-notes/extras/vae/
[4] f_σ is similarly defined.
[5] https://webscope.sandbox.yahoo.com/catalog.php?datatype=l
[6] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html
[7] http://wiki.dbpedia.org/Datasets
[8] http://scikit-learn.org/stable/datasets/twenty_newsgroups.html
[9] https://github.com/unsuthee/VariationalDeepSemanticHashing
[10] STH, LCH, SpH: http://www.dcs.bbk.ac.uk/~dell/publications/dellzhang_sigir2010/sth_v1.zip
[11] http://pixelogik.github.io/NearPy/

	DBPedia					**YahooAnswers** (Yahoo)				
Method	8-bits	16-bits	32-bits	64-bits	128-bits	8-bits	16-bits	32-bits	64-bits	128-bits
LSH [12]	0.0904	0.1040	0.1385	0.1815	0.2506	0.1037	0.1066	0.1107	0.1215	0.1406
SpH [10]	0.4246	0.6083	0.6877	0.6879	0.6787	0.1430	0.1710	0.2204	0.2416	0.2195
LCH [11]	0.2484	0.4556	0.6663	0.7527	0.7907	0.2149	0.3593	0.4559	0.5158	0.5323
STH [7]	0.2381	0.3734	0.5589	0.6903	0.7812	0.2991	0.3808	0.4627	0.5138	0.5304
VDSH [3]	0.6289	0.6951	0.7783	0.8068	0.8588	0.3636	0.4291	0.4611	0.5094	0.5212
NbrReg	**0.6775**$^{\blacktriangle}$	**0.7540**$^{\blacktriangle}$	**0.8250**$^{\blacktriangle}$	**0.8489**$^{\blacktriangle}$	**0.8683**	**0.4179**	**0.4682**$^{\blacktriangle}$	**0.5087**$^{\blacktriangle}$	**0.5298**	**0.5396**
NbrReg+Doc	0.6144	0.7437$^{\blacktriangle}$	0.7996$^{\blacktriangle}$	0.8238	0.8373$^{\triangledown}$	0.4078	0.4565$^{\blacktriangle}$	0.4964$^{\blacktriangle}$	0.5163	0.5255
	20Newsgroups (20NG)					**AgNews**				
LSH [12]	0.0533	0.0558	0.0587	0.0635	0.0720	0.2557	0.2620	0.2713	0.2948	0.3271
SpH [10]	0.0548	0.0567	0.0896	0.1286	0.1218	0.3687	0.4082	0.4359	0.4671	0.5011
LCH [11]	0.1804	0.3075	0.4255	0.4801	0.4688	0.7363	0.7689	0.7789	0.7913	0.7913
STH [7]	0.2337	0.3415	0.4132	0.4340	0.4142	0.6922	0.7541	0.7977	0.8181	0.8243
VDSH [3]	0.2905	0.3125	0.3346	0.3364	0.3874	0.7078	0.7173	0.7471	0.7868	0.8071
NbrReg	0.3463$^{\blacktriangle}$	0.4120$^{\blacktriangle}$	0.4644$^{\blacktriangle}$	0.4768	0.4893$^{\blacktriangle}$	**0.7446**	0.7831$^{\blacktriangle}$	0.8114	**0.8299**	**0.8320**
NbrReg+Doc	**0.3910**$^{\blacktriangle}$	**0.4470**$^{\blacktriangle}$	**0.4898**$^{\blacktriangle}$	**0.5118**$^{\blacktriangle}$	**0.5265**$^{\blacktriangle}$	0.7315	**0.7984**$^{\blacktriangle}$	**0.8149**	0.8233	0.8316

Table 1: Precision of the top 100 retrieved documents on four datasets with different numbers of hashing bits. The bold font denotes the best result at that number of bits. ▲ or ▽ denote the improvement or degradation over the best result of the baselines is statistically significant based on the paired t-test (p-value < 0.05).

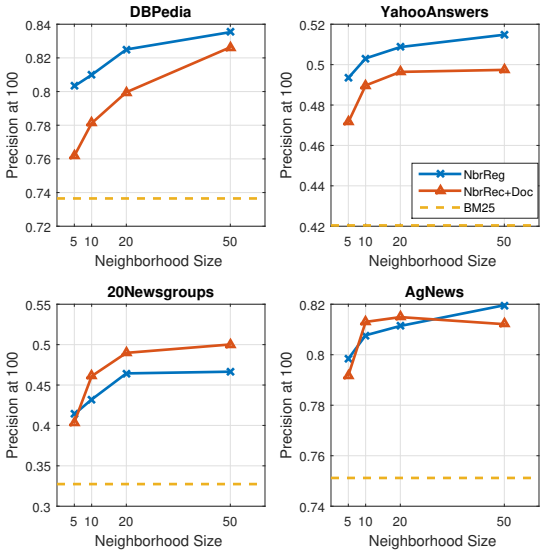

Figure 1: Precision@100 for various neighborhood sizes on the proposed models with the 32-bit hash code.

on most configurations. It shows that the neighbor documents as side information are useful signals for the learning models.

Figure 1 shows the performance of our models as we vary the size of the neighborhood. When we train the models with more neighbor documents, the models generate better binary codes. One explanation is that as we increase the size of neighborhood, we explicitly preserve more information in the document space. However, the performance does not improve after we reach the neighborhood size of 50. We also compare the performance of our models with BM25 model. It shows that our models outperform BM25 on the document similarity retrieval task because both NbrReg and NbrReg+Doc keep only important information via the dimensional reduction while BM25 model keeps all words and contains more noise.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed two deep generative models for text hashing by utilizing weak signals generated by the unsupervised ranking model BM25. Based on the experimental results, we found that explicitly preserving the content of documents and neighborhood is an effective strategy for learning binary codes. In the future, we would like to investigate different kinds of signals from other inexpensive external data sources such as click-through data. Another promising direction is to extend our models for the semi-supervised learning setting.

REFERENCES

[1] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.
[2] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data - a survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.
[3] Suthee Chaidaroon and Yi Fang. Variational deep semantic hashing for text documents. In *SIGIR*, 2017.
[4] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.
[5] Diederik Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.
[6] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on learning to hash. *PAMI*, 2017.
[7] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. Self-taught hashing for fast similarity search. In *SIGIR*, 2010.
[8] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. Neural ranking models with weak supervision. In *SIGIR*, 2017.
[9] Yuanhua Lv and ChengXiang Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *CIKM*, 2009.
[10] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, 2009.
[11] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. Laplacian co-hashing of terms and documents. In *ECIR*, 2010.
[12] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG*, 2004.
[13] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, 2012.
[14] Qifan Wang, Dan Zhang, and Luo Si. Semantic hashing using tags and topic modeling. In *SIGIR*, 2013.