Concept-based Short Text Classification and Ranking*

Fang Wang [†,#1] Zhongyuan Wang [‡,#2] Zhoujun Li [†3] Ji-Rong Wen [‡4]

[†]State Key Laboratory of Software Development Environment Beihang University, Beijing, China

[‡]School of Information Renmin University of China Beijing, China

[#]Microsoft Research Beijing, China

[3]lizj@buaa.edu.cn

[1]wangfang0325@cse.buaa.edu.cn

[2]zhy.wang@microsoft.com

[4]jirong.wen@gmail.com

ABSTRACT

Most existing approaches for text classification represent texts as vectors of words, namely "Bag-of-Words." This text representation results in a very high dimensionality of feature space and frequently suffers from surface mismatching. Short texts make these issues even more serious, due to their shortness and sparsity. In this paper, we propose using "Bag-of-Concepts" in short text representation, aiming to avoid the surface mismatching and handle the synonym and polysemy problem. Based on "Bag-of-Concepts," a novel framework is proposed for lightweight short text classification applications. By leveraging a large taxonomy knowledgebase, it learns a concept model for each category, and conceptualizes a short text to a set of relevant concepts. A concept-based similarity mechanism is presented to classify the given short text to the most similar category. One advantage of this mechanism is that it facilitates short text ranking after classification, which is needed in many applications, such as query or ad recommendation. We demonstrate the usage of our proposed framework through a real online application: *Channel-based Query Recommendation*. Experiments show that our framework can map queries to channels with a high degree of precision ($avg.\ precision = 90.3\%$), which is critical for recommendation applications.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

General Terms

Algorithms, Experimentation

Keywords

Short Text Classification; Query Recommendation; MSN Channel; Taxonomy Knowledge

*This work was done when the first author was an intern in Microsoft Research Asia.

CIKM'14, November 3–7, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2662067.

1. INTRODUCTION

The explosive growth of information makes people feel overwhelmed when browsing for information online. Fortunately, many information technologies have been invented to help. *Text Classification* is one of the key techniques for organizing online information. It has been widely used in News Categorization, Opinion Mining and Spam filtering. Before performing any classification task, text representation is one of the most fundamental tasks [1]. Most well-known techniques [28, 19, 20, 8, 26] represent texts as vectors of terms (words or phrases), namely *"Bag-of-Words"* (BoW).

However, BoW simply looks at the surface word forms and ignores all semantic or conceptual information in the text. Poor performance of BoW is unavoidable in Short Text Classification (STC), because short texts (e.g., search queries, tweets, or Facebook status) are sparse, noisy, and ambiguous. The drawback of surface-based similarity is even more serious in short texts, since two short texts with similar meanings do not necessarily share many words. Most existing work tries to expand the short text by leveraging search engines [31, 11, 37] or external knowledge bases [33, 25, 27] (e.g., ODP, WordNet, and Wikipedia), which needs a time-consuming collection for these expansions. Besides, these expansions are still weak in semantics.

"Bag-of-Words" limits text classification in many applications, especially in short texts or other lightweight online applications that require faster training and new words adaption. As an alternative, we propose representing the short text from a higher perspective of concepts, rather than directly using terms that appear in the text. The notion of "Bag-of-Concepts" is first proposed by Sahlgren, et al. [29]. But the concepts in their paper are some synonym sets or latent dimensions of "Bag-of-Words," not the hyponymy in semantics. Other work defines concepts as units of knowledge, such as entities in WordNet [18] or Wikipedia [14]. For instance, *agriculture*, *agricultural sector* and *agricultural* can be represented by *Agriculture*. Indeed, these concepts can handle problems with synonymy. However, they are still specific representations with limited contribution to semantic similarity. For example, "Jeep" and "Honda" are not synonymy, but they are very similar because they belong to the same concept *Car*.

In this paper, we define a *concept* as a set or class of entities or "things" within a domain[1], such that words belonging to similar classes get similar representations. For instance, "Jeep" and "Honda" can be represented by *Car*. These generic concepts can benefit short text classification. For example, although the two short texts "Beyonce named People's most beautiful woman" and "Lady Gaga Responds to Concert Band" share no common words,

[1]http://www.cs.man.ac.uk/ stevensr/onto/node3.html

they are likely to be the same class of *Music*, since "Lady Gaga" and "Beyonce" can be represented by the same concept *Singer*, which is highly related to *Music*.

There are two advantages of this *"Bag-of-Concepts"* (BoC):

1. *Replace surface matching with semantic similarity*: BoC measures the similarity between short texts from the higher concept level, rather than surface level, which can avoid surface mismatch to some extend.

2. *Tolerance with new terms*: New term arises always but the concept does not. Once the concepts of a category are captured, BoC can handle new words as long as the concept of the word is pre-processed. From this sense, BoC has stronger adaptability in word changing.

We propose a novel framework based on *"Bag-of-Concepts"* for lightweight short-text oriented classification applications (BocSTC). Specifically, given the training texts per class, BocSTC first constructs a concept model for each class, namely *"Bag-of-Concepts"* for each class. During this construction, a large taxonomy knowledgebase is needed to convert terms to concepts. Then, given a short text to be classified, the framework needs to understand its content and associate the short text with the relevant concepts (here we call this operation *Conceptualization* [35]). Based on the conceptual expression, we propose a concept-based similarity mechanism for short text classification and ranking. The framework is suitable for many online lightweight applications, such as query-related applications (e.g., query recommendation) and Ads classification and ranking.

We demonstrate the usage of BocSTC through a real online application: *Channel-based Query Recommendation*. Major Internet portals as MSN and Yahoo! provide diverse channels to facilitate users to browse news by categories. Fig. 1 shows a screenshot for channels on the MSN website.

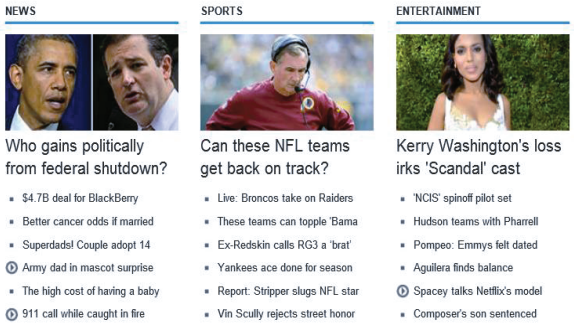


Figure 1: Screenshot for MSN channels

Channel-based Query Recommendation aims to provide the hottest search queries for users when they browse the channels. These queries are recently issued by other people. Given a channel, it will recommend highly related queries, taking into account the query diversity and interestingness. Fig.2 shows a screenshot of the Channel-based Query Recommendation. There are two key tasks in this application:

1. *Mapping query to channel*: Two challenging issues exist in this task: 1) Unlike traditional categories, channels tend to be more general and the content in channels are changing fast along with constantly updated hot news. 2) Queries are very short and difficult to classify using a traditional classifier based on BoW.

Figure 2: Query recommendation for Channel *Living*

2. *Query ranking*: For queries to be recommended, how to measure the relatedness of queries and diversify the recommendations are key issues. Further more, the requirement of quick response for online systems requests the ranking algorithm should be lightweight and practical.

BocSTC represents channels and queries as vectors of concepts, which can simplify the above issues to a large extent. Compared with traditional text classification methods (e.g., SVM), fewer texts are needed for building the concept model for each channel. This saves us much time for manually labeling large numbers of training data and makes it possible to update the learnt model quickly. *"Bag-of-Concepts"* also makes the learnt models tolerance with word changing. For query ranking, BocSTC measures the semantic similarity between the query and channel, which can avoid surface mismatching. Regarding the concepts as subtopics, it is able to diversify the recommendations directly from the subtopic level, without extra query clustering process. Comprehensive experiments are designed to prove its high quality of query recommendations. The techniques we describe in this paper is in production for query recommendation.

The main contributions of this paper are as follows:

- We propose using "Bag-of-Concepts" as a replacement of "Bag-of-Words" in short text representation. This enables us to compute short text similarities at the semantic level of concepts.

- Based on "Bag-of-Concepts," we propose a novel framework for short text classification (BocSTC). Our framework is much more applicable for online lightweight applications, because: i) "Bag-of-Concepts" enables it to adapt term mismatching and word changing; ii) It is a fast learner, since fewer training datasets are required to learn the concept models.

- We demonstrate the usage of BocSTC through a real lightweight application: *Channel-based Query Recommendation*. Experiments show that our framework can map queries to channels with high precision ($avg.\ precision = 90.3\%$). This is very critical for recommendation application.

The rest of the paper is organized as follows. Section 2 briefly introduces the knowledgebase we use for concept model generation. Section 3 first gives the whole picture of BocSTC and then describes its main modules in detail. Section 4 demonstrates the usage of BocSTC in *Channel-based Query Recommendation*. Experiments are presented and analyzed in Section 5. We discuss related work in Section 6 and conclude in Section 7.

2. PRELIMINARIES

In this section, we describe a large knowledgebase that we use to transform "Bag-of-Words" to "Bag-of-Concepts." We also introduce two useful functions to better capture concept information from the knowledgebase, namely *Typicality* and *Concept Cluster*.

2.1 Knowledgebase

We can learn the concepts of words by leveraging existing large-scale knowledgebases, such as Wikipedia, Yago [36] and Probase [40]. We will take Probase as a running example in this paper. Definitely, our techniques can be applied to other knowledgebases. Probase is a probabilistic semantic network that contains millions of concepts. It is rich enough to cover a large proportion of concepts about worldly facts. Terms in Probase are connected by a variety of relationships. Here, we focus on the *Is-A* relationship (although other relationships such as *AttributeOf* [21] are also important to conceptualization). The version of Probase [2] we use contains almost 2.7 million concepts and 4.5 million *Is-A* relationships. For example, *"robin" is-a bird*, and *"penguin" is-a bird*.

The concepts in Probase are fine-grained, which can increase the capacity to distinguish between close classes. For example, given "Angelina Jolie," Probase will return many fine-grained concepts such as *actress*, *Hollywood star* and *movie star*. For "Beyonce," it returns *pop star*, *famous singer* and *musician*. Other knowledge bases do not have such a huge fine-grained concept space. Most of the time, they can map both of "Angelina Jolie" and 'Beyonce" to *celebrity*. But *celebrity* is too general to discriminate between the two close classes -"Movie" and "Music."

2.2 Typicality

For mapping instances to the concept space, we use a probabilistic way to measure the *Is-A* relations, called *Typicality*. For example, given an instance e, which has *Is-A* relationship with concept c, *Typicality* $P(c|e)$ is given by Eq. 1, reflecting how typical of c is among all concepts that contain instance e.

$$P(c|e) = \frac{n(e,c)}{n(e)}, \quad P(e|c) = \frac{n(e,c)}{n(c)} \tag{1}$$

where $n(e,c)$ denotes the co-occur frequency of e and c. $n(e)$ and $n(c)$ are the frequencies of e and c occur during their extraction. Similarly, $P(e|c)$ can measure how typical or popular e is when given c. In our framework, we leverage the *Typicality* to select typical concepts for instances in concept model construction.

2.3 Concept Cluster

Among the large amount of concepts in Probase, many concepts are similar to each other, such as "country" and "nation," "music star" and "pop star," etc. We use *Concept Clusters* to gather similar concepts together, by using a k-Medoids clustering algorithm proposed by Li et al. [22]. One concept cluster can represent one sense or a general topic, recognized with its center concept. For example, for the cluster centered around *country*, most of its members are highly related to *country*, such as *nation*, *asian country*, *developing country*, *region* etc. In this paper, we use the concept cluster in multiple-ways including sense detection in short texts and subtopic representation in the targeted categories.

3. THE FRAMEWORK

In this section, we present the novel framework (BocSTC) for lightweight short-text oriented classification applications (Fig. 3). It consists of two components: offline learning and online classification.

The offline component aims to learn a concept model with good distinguishing power for each target class. Specifically, given the training data for a class i, it first extracts entities from the text, then generates concept candidates by mapping each entity to a concept set. A large taxonomy knowledgebase is exploited for the mapping. A more challenging task is to select representative concepts. The selected concepts should not be too general or too specific for representing class i. We propose weighting the candidates with the combination of entity idf value, concept idf value and typical probabilistic $p(c|e)$, and filter concepts with low weight score. This ranking mechanism is also able to reduce the noise caused by entity-concept mapping. More details are given in subsection 3.1.

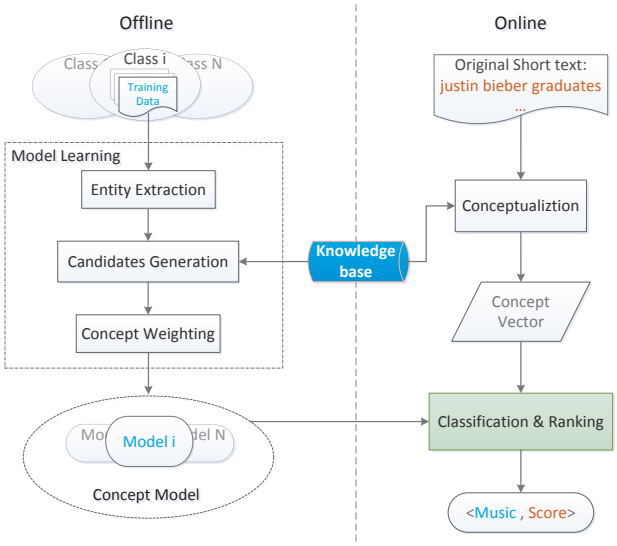


Figure 3: The framework of concept-based short-text classification and ranking

At runtime, when a short text is coming, we need to first understand its main topics by leveraging conceptualization. This enables us to translate the short text to "Bag-of-Concepts," so as to classify it in the same concept level as the concept models.

However, entity disambiguation is a major challenge to accurate conceptualization. Subsection 3.2 shows the details of conceptualization with disambiguation. Based on "Bag-of-Concepts," we propose a similarity-based mechanism to classify short texts. One advantage of this approach is that we can rank the short texts within a class directly using the similarity score, which is convenient for many applications with classification and ranking requirement, such as ads recommendation. Subsection 3.3 explains this approach in detail.

Finally, the ranked results are returned with similarity scores.

3.1 Concept Model Learning

Given training data $D_l = \{d_i, i = 1, 2, .., N\}$ for the class CL_l, we learn its concept model by leveraging the large knowledgebase Probase. The learnt concept model is represented as a concept vector $CM_l = (\langle c_1, w_1\rangle, ..., \langle c_i, w_i\rangle, ..., \langle c_K, w_K\rangle), i = 1, 2, ..., K$, where w_i denotes the weight of concept c_i in class CL_l. The weight could reflect the representative strength of concepts within a class. Specifically, this process is divided into three subtasks: *Entity Recognition*, *Candidates Generation* and *Concept Weighting*.

3.1.1 Entity Recognition

To gain concept expression, we first need to detect the entities in the text so as to access concepts through entities. During the recognition, documents are first split to sentences, and then *Backward Maximum Matching* is used to detect the entities from each sentence. The version of Probase we use contains about 8.26 million instances (e.g., "Beyonce," "Lady Gaga," and "Barack Obama"). Thus, we use all instances in Probase as the matching dictionary.

[2]Probase data is publicly available at http://probase.msra.cn/dataset.aspx

Stemming is also performed to assist in the matching process. After recognition, the extracted entities are merged together and weighted by idf based on different classes. Those with low idf value are removed, because low idf reflects that these entities have low class distinguishing ability.

3.1.2 Candidates Generation

We generate concept candidates from all the extracted entities by leveraging the large amount of *Is-A* relations in Probase. Given entity e_j, we select its top N_t concepts ranked by the *Typicality* $P(c|e)$ (Eq. 1), as its typical concepts. The N_t is usually in the order of tens (in this paper N_t = 20). We merge all the typical concepts as the primary candidate set, and then clean it in the following two ways:

(1) removing stop concepts[3], which tend to be too general to represent a class, such as *item* and *event*;

(2) computing the idf value for each concept in the class level, and removing concepts with low idf value.

Although we remove many non-representative concepts, the noise concepts still exist because simply selecting the top N_t concepts is incapable of processing ambiguous entities. For example, given entity "python" in class *Technique*, our mapping method will result in its top N_t concepts list including *animal*, which is a noise to class *Technique*. We leave this problem to the next subtask.

3.1.3 Concept Weighting

In this subtask, we weight the candidates to measure their representative strengths for each class. Specifically, for each candidate c_k, we aggregate the weights of votes to it from all its entities in CL_l, as its weight in CL_l. Meanwhile, the idf value of c_k and the typicality $P(c_k|e_j)$ is also considered, as Eq. 2 shows:

$$w(c_k, CL_l) = \sum_{e_j \in CL_l} P(c_k|e_j) \times idf(e_j) \times idf(c_k) \qquad (2)$$

According to the weights, we rank the candidate concepts. The noise concepts brought by the *Candidates Generation*, tend to have low ranking scores, since there are few supporting entities for them in CL_l. Thus, concepts with low weights are removed. Finally, we get the concept model CM_l for the class CL_l.

3.2 Short Text Conceptualization

Short Text Conceptualization aims to abstract a set of most representative concepts that can best describe the short text [35, 39]. To avoid over abstracting, specific entities are preferred during entity recognition from the short text. Therefore, we first detect all possible entities and then remove those contained by others. For example, given the short text "windows phone app," the recognized entity set will be {"windows phone," "phone app"}, while "windows," "phone," and "app" are removed. The entity set is then used to conceptualize the short text.

Given the entity list $E_{st_i} = \{e_j, j = 1, 2, ..., M\}$ for a short text st_i, we conceptualize them by leveraging the tens of millions of concept-instance pairs in Probase. Song, et al. [35] estimate the probability of concepts using a naive Bayes model. However, they do not refer to entity disambiguation, which is a key issue affecting the conceptualization accuracy. A typical example is to understand the short text *"apple ipad"*, where "apple" has two senses, namely a famous *Company* and a kind of *Fruit*. In this paper, we use the context "ipad" to assist mapping "apple" to *Company*. We observe that the context has the ability to disambiguate a vague term. For example, when "china" and "jordan" occur together, "jordan" tends to be a *Country*, while when it appears with "nike," it is more likely to be a"brand," because a vague term tends to have the same sense with its near context. Formally, we conceptualize the short text st_i in the following two steps:

Sense Detection: This step aims to detect different senses for each entity in E_{st_i}, so as to determine whether the entity is ambiguous. Probase provides concept clusters to gather similar concepts within a same sense. Thus, we detect the senses of entity e_j directly using the concept clusters of its typical concepts. Denote $C_{e_j} = \{c_k, k = 1, 2, ..., N_t\}$ is e_j's typical concept list and $CCl_{e_j} = \{ccl_m, m = 1, 2, ...\}$ is e_j's concept cluster set. We estimate the ambiguity of e_j by its entropy of concept cluster distribution (Eq. 3).

$$H(e_j) = -\sum_{ccl_m \in CCl_{e_j}} P(ccl_m|e_j) \times log_2 P(ccl_m|e_j) \qquad (3)$$

where $P(ccl_m|e_j)$ denotes the probability of e_j belonging to cluster ccl_m, which is estimated by aggravating the typicality scores for all its concepts belonging to ccl_m. An entity with high entropy value tends to have high uncertainty of cluster distribution.

Disambiguation: Given the entity list E_{st_i} and the identified vague entities, we disambiguate vague entity by leveraging its unambiguous context entities. Denote the vague entity as e_i^v, and unambiguous entity e_j^u. For each cluster of e_i^v, we re-weight them with Eq. 4

$$P^{'}(ccl_m|e_i^v) = \sum_{ccl_n \in CCl_{e_j^u}, e_j^u \in E_{st_i}} P(ccl_m|e_i^v) P(ccl_n|e_j^u) CS(ccl_m, ccl_n) \quad (4)$$

where$CS(ccl_m, ccl_n)$ denotes the concept cluster similarity, calculated with Eq. 5.

$$CS(ccl_m, ccl_n) = \frac{1}{|ccl_m|} \sum_{c_k \in ccl_m} Max_{c_j \in ccl_n} \frac{|E_{c_k} \cap E_{c_j}|}{|E_{c_k} \cup E_{c_j}|} \qquad (5)$$

where E_{c_k} is the entity list including typical entities that belong to concept c_k. We reserve the $ccl^*_{e_i^v}$ that has the most probability as e_i^v's sense. For unambiguous entities, we choose their dominant clusters marked as $ccl^*_{e_j^u}$. Then we merge all clusters extracted from all entities in E_{st_i} as $\{\bigcup_j^M ccl^*_{e_j}\}$. Each concept cluster $ccl^*_{e_j}$ is represented as a concept vector marked as C_j, where each component is a Probase concept valued with $P(c|e_j)$, so that each short text is conceptualized in the same concept space as the concept models.

3.3 Classification and Ranking

In this subsection, we describe the similarity-based mechanism to classify the given short text st_i and rank the items assigned to a class CL_l.

3.3.1 Classification

The intuition of classification is simple: classify the short st_i to the class CL_l that is most similar with st_i based on the same concept space. Given the concept model CM_l for class CL_l, and st_i's concept expression $C_{st_i} = \{C_j, j = 1, 2, ..., M\}$, we measure the similarity between st_i and CL_l with Eq. 6.

$$Sim(st_i, CL_l) = \sum_{C_j} \sum_{c_k \in C_j \wedge c_k \in CM_l} P(c_k|e_j) \times w(c_k, CL_l) \qquad (6)$$

[3]Stop concepts generally have many diverse instances and tend to be in the high level of concept hierarchy. These concepts are already pre-recognized with these two rules.

We select CL_l^* (Eq. 7) that has the maximum similarity with st_i as the classification result. The max similarity score is assigned to st_i as its weight in CL_l^*.

$$CL_l^* = \underset{CL_l \in CL}{\operatorname{argmax}} Sim(st_i, CL_l) \quad (7)$$

A possible adjustment to this method is that one can define a threshold for each class to further refine the classification. For applications with high accuracy requirement, it is necessary to filter out the items with low maximum similarity score.

3.3.2 Ranking

Apart from classification, many applications require ranking their items in the meanwhile, such as channel-based query recommendation, advertisement matching for some topics, etc. Our concept-based classification framework has advantages for ranking classification results in a class. Specifically, two ranking mechanisms are proposed to meet with two typical ranking requirements: Ranking by *Similarity* and Ranking with *Diversity*.

Ranking by Similarity: This is the most common way to rank items. For example, in an ads matching problem, search engines rank bid keywords by their similarities to the user query. In our framework, as each short text st_i assigned to CL_l has a similarity score, we can rank them directly by their scores in descending order. Based on the simple ranking mechanism, items with more typical concepts of CL_l tend to have higher ranked positions. Based on concept level similarity, term mismatching can be tackled to some extant.

Ranking with Diversity: Diversity is an important feature for recommending related applications, which affects the *User Experience* directly. Most existing approaches [12, 23, 16] diversify the recommendations based on different aspects or subtopics of the recommended target. Generally, a clustering or other subtopic mining process is needed to generate different aspects. While in our framework, this step is no longer needed because we can directly gain class aspects by leveraging Probase concept clusters. This makes our framework more adaptive to real-time ranking applications. Regarding the concept clusters as subtopics of the class, we can diversify the short texts by subtopic *Proportionality* proposed by Dang et al [12].

4. CHANNEL-BASED QUERY RECOMMENDATION

In this section, we demonstrate the usage of BocSTC through a real application: Channel-based Query Recommendation.

This online application aims to anticipate user search needs when browsing different channels, by recommending the hottest and highly related queries for a given channel. There are three difficulties in this application: i) the recommended target (Channel) is "short," lack of training data or any user preference logs; ii) a need to understand the short text (user query); iii) requiring both classification and ranking (how to identify which channel the query belongs to and how to rank the recommendations with diversity). These difficulties also indicate the application scenarios of our proposed framework. The following three subsections describe how we tackle these issues using BocSTC.

4.1 Channel Preference Generation

The recommended target - channel tends to be a general category, such as *Living*, *Money*, or *Entertainment*. Unlike traditional recommendation system, we have no log data as query click data or user reading preferences (article click data) in the given channel. It is not practical to label the training samples artificially. Moreover, the content in the channels is changing fast along with constantly updated hot news. This puts forward a higher requirement of adapting word changing. A more promising way is to seize the core topics hidden behind the surface words appeared in the channel.

BocSTC tries to capture the typical concepts from the hot news crawled from pages listed in each channel. We just need to process the titles to get typical concepts, since the article title can reflect the main topics of the article. This saves us much time to process a large amount of long texts, so as to enable the learnt concept models to adapt quick updating. Table 1 gives some typical examples of the learnt concept model. It shows the learnt typical concepts in different concept clusters (topics) for *Channel Music*.

Table 1: The Learnt Concept Model for "Channel Music"

Typical concepts of different topics		
Singer:	*Song*:	*Instrument*:
performer	good song	musical instrument
pop star	classic song	electronic instrument
pop artist	hip-hop song	string instrument
Music:	*Band*:	*Musician*:
music style	rock band	guitarist
musical genre	metal band	guitar player
musical form	pop band	pianist

4.2 Query Conceptualization

In order to map the query to an appropriate channel, we first need to know what the query is talking about, namely understanding the search intent of the user issued the query.

Understanding query intent is one of the most basic components of information retrieval systems [10], which has been studied extensively in recent years. Most existing approaches utilize machine learning techniques to predict the user's intent. However, it often needs to enrich query features through search engines [31, 11] or utilize log data (i.e., query log and search-click-through log) [4, 24] as unlabeled data for training a query classifier. For mapping queries to appropriate channels, these methods are not desirable. First, enriching a query through search engines is too time-consuming for a real-time online application. Second, there are not sufficient training samples for training a query classifier using traditional statistic machine learning methods. Manual labeling is unpractical in this application.

With very little human effort, BocSTC captures query topics from the concept level, by leveraging the short text conceptualization module mentioned in subsection 3.2. Query conceptualization infers typical concepts from a set of instances detected from the query text, so as to map the query to an appropriate channel in the same concept space as the learnt concept models. During this process, we also consider the entity disambiguation by using its context entities in the query. Table 2 shows some examples of our query conceptualization.

Table 2: Examples of Query Conceptualization

Query	**Detected Entity**: **typical Concepts**
apple engineer	*apple*: company, corporation, firm *engineer*: professional, expert, occupation
the temptations	*temptation*: artist, popular artist, entertainer
george clooney dated lucy liu	*george clooney*: celebrity, movie star, actor *lucy liu*: celebrity, star, asian actress

4.3 Query Recommendation for Channels

In this subsection, we describe the usage of our *Classification and Ranking* module customized for channel-based query recommendation.

After the former two steps, both of the channels and candidate queries to be recommended are represented as "Bag-of-Concepts." Based on the same concept space, we first classify query candidates with the similarity-based method mentioned in subsection 3.3.1. We then recommend queries with the diversity ranking method proposed in subsection 3.3.2. The queries classified to the channel are first ranked by their *similarity* scores, and then the top N (i.e., $N = 100$) queries are reserved for diversity recommendation. This is done to filter non-typical queries classified to the channel, so as to improve the quality of the recommendation.

We use the *proportionality*-base algorithm PM-2 [12] to diversify the recommendations, where the concept clusters are used as the channel aspects. PM-2 considers a result list most diverse, with respect to some set of topics related to the recommended target, when the number of queries it provides on each topic is proportional to the topic's popularity. Given the list R of the top N queries in channel CL_l, the seat number N_s(we generally set N_s to 5, 10 or 20) and the topic list $CCL_l = \{ccl_i, i = 1, 2, ...M\}$ in the channel. It first selects a topic with the most quotient and then assigns a relevant query to the selected topic ccl_i^*. It iterates these two steps until all the seats are sold out. In the second step, it considers other topics as well (Eq. 8).

$$q^* = \underset{q_j \in R}{\operatorname{argmax}}\{\lambda qt[i^*] \times P(q_j|ccl_i^*) + (1-\lambda) \sum_{i \neq i^*} qt[i] \times P(q_j|ccl_i)\} \tag{8}$$

Where $qt[i]$ denotes the quotient of the topic i and $P(q_j|ccl_i)$ denotes the probability that the query q_j belongs to cluster ccl_i. λ is a parameter for tuning the weight between ccl_i^* and other topics.

PM-2 focuses more on the diversity of the topic level, while the surface word level is neglected, which also has a significant impact on the user experience. For example, given the former recommendation *"elton john worries lady gaga,"* recommending *"elton john lady gaga health"* will damage the user experience. In this paper, to avoid repeating recommendations, we introduce a word distance factor when assigning a query to ccl_i^* (Eq. 9)

$$q^* = \underset{q_j \in R}{\operatorname{argmax}}\{Score_{PM\text{-}2} \times \textbf{Distance}(q_j)\} \tag{9}$$

Where we abbreviate $Score_{PM\text{-}2}$ to the original PM-2 score in Eq. 8 due to the space limitation. $Distance(q_j)$ denotes the minimum word distance of q_j to the selected queries that already have a seat(Eq. 10).

$$Distance(q_j) = \underset{q_i \in S'}{\operatorname{argmin}}\{1 - \frac{|W_{q_i} \cap W_{q_j}|}{|W_{q_i} \cup W_{q_j}|}\} \tag{10}$$

Where $S^{'}$ is the set of queries that already have a seat and W_{q_i} denotes the word set of q_i.

5. EXPERIMENTS

In this section, we evaluate the performance of BocSTC on the real application - *Channel-based query recommendation*. The experiments are divided into two evaluation parts: *Query Classification* and *Result Diversity*. We first introduce the datasets and then present experimental results to assess the effectiveness of our proposed BocSTC.

5.1 DataSet

Four commonly used channels are selected as our targeted channels: *Money*, *Movie*, *Music* and *TV*. Note that the last three ones are somewhat similar to each other. We select them with the aim of testing the classification performance of similar categories. We now introduce the real-world training and test datasets used in our experiment.

Training dataset: Given the hot news crawled from pages listed in each channel, we randomly select 6,000 items for each channel, as Tab. 3 shows. We then extract the title and body text from each article. The titles are used as training data for BocSTC. The body texts together with the titles are used for training other baseline models. From this angle, we can see that much fewer texts are needed in BocSTC.

Table 3: Training Set Constitution

Channel	# Articles	# Sample
Money	180,610	6,000
Movie	6,360	6,000
Music	7,623	6,000
TV	12,888	6,000

Test dataset: We use the real queries from a Bing query flow over the course of 5 hours as the source of our test dataset. To alleviate the burden of manual annotation, we first filter these queries with a pre-trained classifier, then annotate the pre-classified queries manually. We obtain 841 labeled queries, from which, 200 are selected randomly for verification and 600 for testing, as Tab. 4 shows. Note that a query may belong to multiple channels.

Table 4: Test Set Constitution

Channel	# Queries	# Verification	# Test
Money	194	50	144
Movie	206	50	152
Music	234	50	152
TV	207	50	152
Total #	841	200	600

Unlike articles in the training dataset, queries are generally very short. To gather more information about the short queries for baseline methods, we expand queries by leveraging popular search engines. According to Dou Shen, et al. [33], we use the snippets of the top 40 returned pages to represent it. We remove stop-words and perform stemming for all the data. Some statistical numbers from the training and test data are listed in Tab. 5.

Table 5: Length of Experimental Data (# of unigrams)

Data Type	Max	Min	Ave.
Query	9	2	2.93
Article Title	27	2	8.58
Article	225	68	109.2
Expanded Query	432	101	245.05

5.2 Query Classification Performance

In this subsection, we present the query classification performance of our model on each channel.

5.2.1 Experimental Setup

In order to demonstrate the effectiveness of BocSTC, we compare it with the following methods:

(1) *A naive entity-based method (Entity_ESA)*. Many Wikipedia-based work [15, 41] has been devoted to measuring semantic relatedness. To handle the surface mismatch in the word level, we leverage the state-of-the-art relatedness method - ESA [15] to map the query to the its most similar channel. To better capture the meaning words, we parse the test queries and channel titles in the training dataset to entities with the same method in section 3.1.1. We compute the semantic similarity between

query Q_i and channel CL_l by Eq 11.

$$Sim(Q_i, CL_l) = \sum_{e_i \in Q_i} \sum_{e_j \in CL_l} ESA(e_i, e_j) \times idf(e_j) \tag{11}$$

To simplify the computing, we select the top 500 entities (ranked by idf) for each channel as their representative entities. The channel that has the maximum similarity with Q_i is regarded the classification result. One can see that the Entity_ESA is similar with our BocSTC. The main difference is that we use "Bag-of-Concepts" rather "Bag-of-Entity" in BocSTC.

(2) *Vector Space Model (VSM)* [30]: VSM is an algebraic model for representing text documents as vectors of terms, which is commonly used in relevance rankings. Here we use it as a basic method for mapping queries to channels. A straightforward approach is to leverage query snippets by VSM. We denote the expanded query as Q^e, the target channel as CH and the article in each channel as D respectively. For each article D_j, we can compute its similarity with the query Q_i^e based on their word vectors. After that, the channel CH^*, to which the most similar article D^* belongs, is regarded as the classification result.

$$D^* = \underset{D_j}{\operatorname{argmax}}\{Similarity(Q_i^e, D_j)\} \ \& \ D^* \in CH^* \tag{12}$$

(3) *Language Model (LM)* [34]: It is used in a similar way as VSM - selecting the rank 1 *"result."* The probability of a query Q_i being "generated" by a document D_j is denoted by $P(D_j|Q_i)$, which from Bayes' formula is given by

$$P(D_j|Q_i) \propto P(Q_i|D_j)p(D_j) \tag{13}$$

where $p(Q_i|D_j)$ is the query likelihood given the document D_j and $P(D_j)$ is the prior belief that D_j is revelent to any query. In most existing work, $p(D_j)$ is assumed to be uniform, so it does not affect the ranking. By using a unigram model, Eq.13 can be reformulated as follows:

$$P(D_j|Q_i) \propto P(Q_i|D_j) = \Pi_{k=1}^{n} P(q_k|D_j) \tag{14}$$

where q_k is the k^{th} term of query Q_i. By using LM, we first select the article D^* that is most possible to generate Q_i, and take D^*'s channel as the classification result CH^*. That is:

$$D^* = \underset{D_j}{\operatorname{argmax}}\{P(Q_i|D_j)\} \ \& \ D^* \in CH^* \tag{15}$$

In this paper, we build two language models: one is for the article level and the other for channel level, marked as LM_d and LM_{ch} respectively. Considering the model level, in query expression, we use the original query Q in LM_d and the expanded version Q^e in LM_{ch}.

(4) *Support Vector Machine (SVM)* [8]: SVM is a kind of statistical classifier, used as a base classifier for query categorization in the KDDCUP 2005 winning solution. In this paper, LibSVM [6] is used to train a SVM classifier based on our training dataset.

Among the baseline methods, in VSM, $Cosine$ is used as the similarity measurement and the normalized TF is used for the dimension value. In estimating $p(D_j|Q_i)$ for LM, we use the Dirichlet smoothing method proposed by C. Zhai [42]. There is a Dirichlet Prior μ to be set. After turning, we set μ to 500 for LM_d and 2000 for LM_{ch}. In training the SVM classifier, we verify the classifier with the query verification dataset. Also, the normalized TF is used as the value on each dimension.

5.2.2 Performance Evaluation

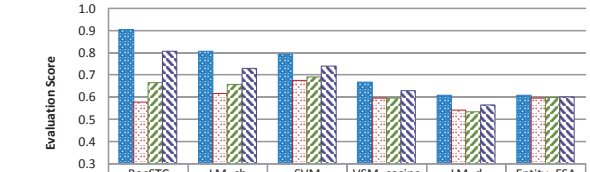
In this evaluation, the $Precision$, $Recall$, $F\text{-}value$ are employed as the performance metrics to evaluate the quality of Query Classification. The $F\text{-}value$ is the harmonic mean of precision and recall. F_1 is commonly used when the $precision$ and $recall$ are evenly weighted. While in the case of emphasizing $precision$, $F_{0.5}$ is used. We use both of them for better evaluation.

Experimental results of the baseline methods on the real query test dataset are respectively plotted in Fig. 4. For the four metrics, the vertical axis represents the average score of the four channels. From this figure, we draw the following observations:

(1) *BocSTC* performs much better than other baselines in terms of $precision$, nearly 10 percentage points (9.73%) higher than the runner-up - LM_{ch}. Although its *Recall* is less than satisfactory, it gets the highest $F_{0.5}$ (80%+). This is acceptable in the lightweight application - called *Recommendation Queries for Channels in Portal Websites*, which pays much more attention to the *precision*.

(2) Specifically, the methods LM_{ch}, SVM and VSM use the expanded query expression, while LM_d does not. We can see that the performance of LM_{ch} is significantly better than LM_d. This suggests query expansion plays an important role. However, query expansion is infeasible for online applications that require efficient timeliness. But our *BocSTC*'s result are comparable to LM_{ch} without query expansion. This demonstrates the significance of our proposed lightweight classifier by exploiting explicit concepts.

(3) Entity_ESA also attracts our attention because it gets the worst results. This is not surprising since it is a very simple distance model even though ESA is used for computing the similarity. However, it shows the power of "Bag-of-Concepts" from another angle. In fact, the classification part in BocSTC is also a simple distance model. But by using the BoC in replacement of BoW, it performs the the best in this task.

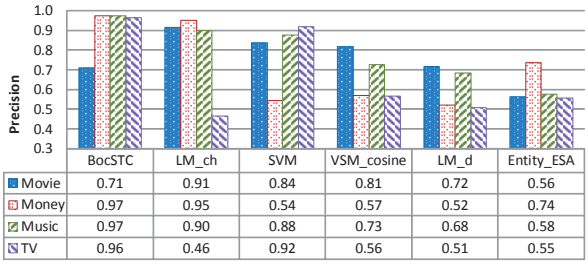


	BocSTC	LM_ch	SVM	VSM_cosine	LM_d	Entity_ESA
Avg. Precision	0.90	0.81	0.79	0.67	0.61	0.61
Avg. Recall	0.58	0.61	0.67	0.59	0.54	0.59
Avg. F1-Score	0.66	0.66	0.69	0.59	0.53	0.60
Avg. F0.5-Score	0.81	0.73	0.74	0.63	0.56	0.60

Figure 4: Performance on query classification

Figure. 5 shows the detailed $precision$ performance of these methods on each channel. We can see that our *BocSTC* has a significant advantage in three channels except for the "Movie" channel. The best performance in this channel goes to LM_{ch}, higher than our model by nearly 20%. This forces us to investigate the result of "Movie Channel" in detail.

We look up the test data and compare it with our predicted ones, listed in Tab. 6. We can see that nearly half of the queries in the *Movie channel* are not assigned to any channel label (73/152), which greatly reduces the $recall$. We believe this is because the entity coverage in Probase is still limited. Since many queries in this channel contain movie titles, such as *"Journey 2 The Mysterious Island."* New film names come out continually. It is barely covered by the knowledgebase in time. However, this can be easily resolved by named entity extraction techniques.



	BocSTC	LM_ch	SVM	VSM_cosine	LM_d	Entity_ESA
Movie	0.71	0.91	0.84	0.81	0.72	0.56
Money	0.97	0.95	0.54	0.57	0.52	0.74
Music	0.97	0.90	0.88	0.73	0.68	0.58
TV	0.96	0.46	0.92	0.56	0.51	0.55

Figure 5: Precision performance on each channel

As can be seen from Tab. 6(b), queries in the *Music channel* make a lot trouble for accurately predicting *Movie* queries. This is because the two channels are similar to each other. In fact, most of the stars have a variety of careers (e.g., actor, singer, host or producer), such as "Beyonce" of America and "Zhao Wei" of China. However, we can make up for this kind of mistake by leveraging the ranking score given by BocSTC.

Table 6: Performance Analysis on "Movie Channel"

(a) Real Queries

Predicted to	# Query
Movie	64
TV	9
Music	5
Money	1
No Label	**73**
In total	152

(b) Predicted Queries

Belong to	# Query
Movie	64
TV	13
Music	**55**
Money	0
In total	132

5.3 Recommendation Evaluation

In this subsection, we evaluate the performance of BocSTC in terms of recommendation interestingness.

5.3.1 Experimental Setup

For each channel, the input of our query ranking module is the queries with scores assigned by classification module in *BocSTC*. In order to demonstrate the effectiveness of the proposed ranking method (section 4.3), we compare it with following methods:

- LM_{ch}: We rank the queries for each channel with their LM score, to see the diversity performance of the LM_{ch} model.
- *SVM*: In this method, each query has a probability of its channel. We leverage this value to rank the queries with same purpose with LM_{ch}.
- *BocSTC-Original*: We use the original query list ranked by similarity as a baseline to see the performance of our diversity recommendation.
- *BocSTC-PM2* [12]: This is the key method in our diversity module. Given the original query list, the PM-2 is used to diversify the queries by subtopic proportionality.

In this evaluation, we set the seat number N_s to 20, namely recommending 20 queries for each channel. For each of the baselines, we select the top 20 queries as a result list to be labeled. Then, we manually annotate these lists with the following guidelines:

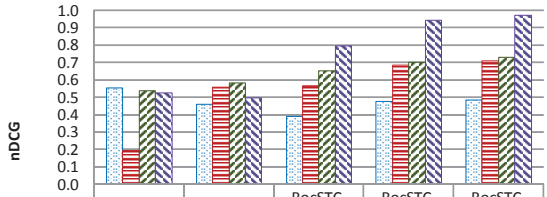
- *Unrelated*: the query is **irrelevant** to the Channel.
- *Related but Uninteresting*: the query is **relevant** to the channel, but it's **uninteresting** or **repeated** in the content mentioned in previous recommendations.
- *Related and Interesting*: the query is **relevant** to the channel and it would **raise interest** in the topics covered by the query.

5.3.2 Performance Evaluation

We use $nDCG$ as the evaluation metric. Since we do not know the ideal DCG in our datasets, we set it to the highest score, which is computed with the Eq. 16:

$$iDCG@k = rel_{ideal} + \sum_{i=2}^{k} \frac{rel_{ideal}}{\log_2 i} \tag{16}$$

where rel_{ideal} is the ideal label, which is 2 in this evaluation. Fig. 6 shows the experimental results, where our refined version of PM-2 is marked as *BocSTC-PM2+*, which combines term diversity with subtopic diversity.



	LM_ch	LibSVM	BocSTC-Original	BocSTC-PM2	BocSTC-PM2+
Channel Money	0.552	0.457	0.389	0.473	0.483
Channel TV	0.191	0.557	0.565	0.682	0.707
Channel Music	0.534	0.580	0.651	0.700	0.729
Channel Movie	0.524	0.493	0.795	0.941	0.970

Figure 6: Diversity performance on each channel

From this figure, we draw the following observations:

(1) By using *PM-2*, the diversity performance of query recommendation is greatly improved in all the four channels (Movie↑ 14.6%, Music↑ 4.9%, TV↑ 11.7% and Money↑ 8.4%). The *BocSTC-PM2+* wins, with the average $nDCG$ further ↑ 2.3%. This demonstrates the significance of our refined diversity method.

(2) Our diversity method is a significant improvement over LM_{ch} and SVM, with the average $nDCG$ up more than 10%. But in *Money channel*, the LM_{ch} performs better. However, the highest score is only 55.2%. After checking the queries in this channel, we found that the top ranked queries did not interest our annotators, partially because most candidate queries share the same topic-*"Bank,"* which makes the recommendation monotonous. Another reason is again the Probase entity coverage. For example, "property brothers" is not contained in Probase, and it is parsed into "property" and "brothers" instead. This makes it most likely to be related to "Money."

5.3.3 Anecdotal Evidence

We present example queries recommended by *BocSTC*, LM_{ch} and *SVM* in Tab. 7.

From the examples of *SVM*, we can see that directly ranking by classification scores can hardly perform well in diversity recommendation. For instance, in the *Movie channel*, "jonny depp" is ranked higher than "depp movie awards" (it is not ranked in the top 5), but the latter seems to be more interesting because it provides extra information. This cannot be handled by traditional text classifiers. As for LM_{ch}, although its top ranked queries are related to the channels, their topics are relatively monotonous. For example, in the *Music channel*, "music youtube," "chris brown music," and "beyonce music" appear in the top 5 at the same time. This is detrimental to the user experience.

The queries recommended by *BocSTC* are more interesting, especially in the *Movie channel* and the *Music channel*, because they cover most subtopics of their channels. In other words, queries reflecting more topics are more likely to be interesting. It also takes

Table 7: Top 5 recommendations for each channel

Channel	BocSTC	LM_{ch}	SVM
Movie	tom hanks rita wilson bosom buddies	2012 mtv movie awards red carpet	dark night rises
	lindsay lohan as elizabeth taylor	depp movie awards	jonny depp
	christian bale gets choked up	film composer morricone	prometheus review
	emma watson 2012	prometheus review	titanic movie
	george clooney dated lucy liu	watch men in black 3	avengers trailer
Music	elton john worries lady gaga	donna summer funeral	carly simon
	the temptations	music youtube	donna summer funeral
	madonna nazi image	chris brown music	elton john worries lady gaga
	sara lownds dylan and bob dylan	mtv music videos	paul simon
	faith hill tim mcgraw married for 15 years	beyonce music	elton john lady gaga health
TV	today show recipes	yolanda adams morning show	secret life of the american teenager
	teen mom sentenced	yard crashers	american horror story season 2
	news channel 8	world trade center	canceled shows 2012
	mtv music videos	workaholics season 3	make it or break it
	master chef 2012	words with friends cheats	master chef 2012
Money	comercia web banking	national grid online bill pay	ally financial gmac
	neighbors credit union	stock market plunge	astoria federal savings
	tn unemployment	union first market bank	central bank
	morgan stanley clientserv	small business marketing	chargeback it
	chase credit cards	bad company	commerce bank online

word distance into consideration, so as to avoid repeating queries in different subtopics but with similar words.

6. RELATED WORK

Techniques proposed in this paper are mainly related to *short text classification* and *query recommendation*.

6.1 Short Text Classification

Short text classification delivers short texts (e.g., queries, tweets and comments) to some pre-defined categories based on content analysis. Most existing approaches are mainly based on feature expanding. Generally, there exists two expanding directions. One is to obtain extra context information through search engines [31, 11, 37]. The expanded short texts are regarded as long texts and can be classified with long text classification approaches [32]. But this expansion is not an ideal solution for some online applications, since it is very time consuming and heavily dependent on search engine quality. The other one is to expand features by leveraging large external knowledgebases such as Wikipedia and WordNet [33, 25, 17, 27]. These methods discover a set of explicit or implicit topics and then connect the short text through these topics. Using pre-defined topics or taxonomy relaxes the dependence on search engines. But its adaptability can be an issue since the pre-defined topics may not be available for certain applications [7], and the topic granularity is hard to be defined.

6.2 Query Recommendation

Query recommendation (QR) is a common tool used by search engines to assist users in searching or browsing information. When generating query recommendations for a user, a natural approach is to leverage the user search session (the user's most recently submitted queries) [43], the clicked documents [9] or other log data [3]. Techniques for QR based on these context information have been studied extensively [38, 13, 2].

However, not all the scenarios of QR have such a sufficient context information, such as the application we mentioned in this paper - *Recommending Queries to Channels*. When there is a lack of user preference or any other query logs for a given target, existing approaches are powerless in recommending high related queries to the targeted object. Bordino et al. [5] try to suggest interesting queries to users when they're reading an article. Similarly, they don't have any user preference. But the long text of the viewed article can be leveraged to recognize the core entities the user is interested in. But for a channel, the preference can not be obtained from one article. In this case, we propose learning the topical preference of the targeted channel from its content, and map a given query to an appropriate channel from the topic level. From this perspective, our work seems to be similar to the issue of Query Classification (QC) [33]. However, QC only assigns a query a category. It does not refer to rank these queries after the classification.

7. CONCLUSION

In this paper, we propose a novel framework for short text classification and ranking applications. Compared with existing approaches for short text classification, our framework has two advantages: i) It measures the semantic similarities between short texts from the angle of concepts, so as to avoid surface mismatch. ii) Fewer training data are needed to learn the concept model per class, since few terms together are able to reflect one concept. These advantages make it suitable for online lightweight applications that need to deal with short texts with requirements of fast learning and word changing adaption. We demonstrate the usage of our proposed framework through a real online application: *Channel-based Query Recommendation*. The experimental results show that our method can significantly improve classification precision by 9.73% and also perform well on diversity recommendation.

There is also much future work. For example, we can further improve it by exploiting the similarities between concepts. Besides, the classifier/ranking in current framework is actually a simple distance model. It will be very interesting to incorporate powerful machine learning techniques on top of the "Bag-of-Concepts" vectors to improve the discrimination effectiveness.

8. ACKNOWLEDGMENTS

This work was supported by NSFC (Grand Nos. 61170189, 61370126, 61202239), the Research Fund for the Doctoral Program of Higher Education (Grand No. 20111102130003), the Fund

of the State Key Laboratory of Software Development Environment (Grand No. SKLSDE-2013ZX-19), and Microsoft Research Asia Fund (Grand No. FY14-RES-OPP-105). This work was partially supported by the National Key Basic Research Program (973 Program) of China under grant No. 2014CB340403 and the National Natural Science Foundation of China under grant No. M13210007.

9. REFERENCES

[1] C. C. Aggarwal and C. Zhai. *Mining text data*. Springer, 2012.

[2] A. Anagnostopoulos, L. Becchetti, C. Castillo, and A. Gionis. An optimization framework for query recommendation. In *WSDM*, pages 161–170. ACM, 2010.

[3] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *EDBT*, pages 588–596. Springer, 2005.

[4] S. M. Beitzel, E. C. Jensen, O. Frieder, D. D. Lewis, A. Chowdhury, and A. Kolcz. Improving automatic query classification via semi-supervised learning. In *ICDM*, 2005.

[5] I. Bordino, G. De Francisci Morales, I. Weber, and F. Bonchi. From machu_picchu to rafting the urubamba river: anticipating information needs via the entity-query graph. In *WSDM*, pages 275–284. ACM, 2013.

[6] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *TIST*, 2(3):27, 2011.

[7] M. Chen, X. Jin, and D. Shen. Short text classification improved by learning multi-granularity topics. In *IJCAI*, pages 1776–1781. AAAI Press, 2011.

[8] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[9] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR*, pages 239–246. ACM, 2007.

[10] W. B. Croft, M. Bendersky, H. Li, and G. Xu. Query representation and understanding workshop. In *SIGIR Forum*, volume 44, pages 48–53, 2010.

[11] H. K. Dai, L. Zhao, Z. Nie, J.-R. Wen, L. Wang, and Y. Li. Detecting online commercial intention (oci). In *WWW*, 2006.

[12] V. Dang and W. B. Croft. Diversity by proportionality: an election-based approach to search result diversification. In *SIGIR*, pages 65–74. ACM, 2012.

[13] H. Feild and J. Allan. Task-aware query recommendation. In *SIGIR*, pages 83–92. ACM, 2013.

[14] E. Gabrilovich and S. Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *AAAI*, 2006.

[15] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.

[16] J. He, V. Hollink, and A. de Vries. Combining implicit and explicit topic representations for result diversification. In *SIGIR*, pages 851–860. ACM, 2012.

[17] X. Hu, N. Sun, C. Zhang, and T.-S. Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *CIKM*, pages 919–928. ACM, 2009.

[18] L. Huang. *Concept-based text clustering*. PhD thesis, The University of Waikato, 2011.

[19] A. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS*, 14:841, 2002.

[20] Y.-H. Kim, S.-Y. Hahn, and B.-T. Zhang. Text filtering by boosting naive bayes classifiers. In *SIGIR*, 2000.

[21] T. Lee, Z. Wang, H. Wang, and S.-w. Hwang. Attribute extraction and scoring: A probabilistic approach. In *ICDE*, pages 194–205. IEEE, 2013.

[22] P. Li, H. Wang, K. Q. Zhu, Z. Wang, and X. Wu. Computing term similarity by large probabilistic isa knowledge. In *CIKM*, pages 1401–1410. ACM, 2013.

[23] R. Li, B. Kao, B. Bi, R. Cheng, and E. Lo. Dqr: a probabilistic approach to diversified query recommendation. In *CIKM*, pages 16–25. ACM, 2012.

[24] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *SIGIR*, 2008.

[25] Y. Li, D. McLean, Z. A. Bandar, J. D. O'shea, and K. Crockett. Sentence similarity based on semantic nets and corpus statistics. *TKDE*, 18(8):1138–1150, 2006.

[26] H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *SIGIR*. ACM, 1997.

[27] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW*, 2008.

[28] J. R. Quinlan. Induction of decision trees. *Machine learning*, pages 81–106, 1986.

[29] M. Sahlgren and R. Cöster. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *COLING*, page 487. ACL, 2004.

[30] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 1975.

[31] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Q2c@ust: our winning solution to query classification in kddcup 2005. *SIGKDD*, 7(2):100–110, 2005.

[32] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Query enrichment for web-query classification. *TOIS*, 24(3):320–352, 2006.

[33] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *SIGIR*, 2006.

[34] F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM*, pages 316–321. ACM, 1999.

[35] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen. Short text conceptualization using a probabilistic knowledgebase. In *IJCAI*, pages 2330–2336. AAAI Press, 2011.

[36] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706. ACM, 2007.

[37] A. Sun. Short text classification using very few words. In *SIGIR*, pages 1145–1146. ACM, 2012.

[38] I. Szpektor, A. Gionis, and Y. Maarek. Improving recommendation for long-tail queries via templates. In *WWW*, pages 47–56. ACM, 2011.

[39] Z. Wang, H. Wang, and Z. Hu. Head, modifier, and constraint detection in short texts. In *ICDE*, pages 280–291, 2014.

[40] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, pages 481–492. ACM, 2012.

[41] E. Yeh, D. Ramage, C. D. Manning, E. Agirre, and A. Soroa. Wikiwalk: random walks on wikipedia for semantic relatedness. In *ACL Workshop*, pages 41–49. ACL, 2009.

[42] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342. ACM, 2001.

[43] Z. Zhang and O. Nasraoui. Mining search engine query logs for query recommendation. In *WWW*, pages 1039–1040, 2006.