Deep Classifier for Large Scale Hierarchical Text Classification

Dingquan Wang, Weinan Zhang, Gui-Rong Xue, and Yong Yu

Dept. of Computer Science and Engineering, Shanghai Jiao Tong University,
Dongchuan Road. 800, 200240 Shanghai, China
{dqwang,wnzhang,grxue,yyu}@apex.sjtu.edu.cn

Abstract. In this competition, we refined a novel algorithm for classification on large scale documents with deep category structure based on a two-stage strategy known as the deep-classifier [1]. The basic idea of deep-classifier is to take advantage of the better performance of k-NN relevance search on large scale categories and the higher precision of the Naive Bayes for multi-class classification. As a result, it achieved improvement on both performance and efficiency in our experiment.

1 Introduction

In our algorithm, the training set was firstly well pruned using search method before the second stage's accurate and efficient classify. There are two issues mainly affect the performance of our classifier, the first is the large category and feature set for each document which seriously slow down the execution time, second is the deep category structure that decrees the accuracy of the result. To deal with the first problem, feature selection was carefully performed based on the vector space model of tf×idf. To deal with the second problem, we came up to a bottom up refining method by adding in the hierarchy information. Since the two stages are different evaluation aspects of the same category, the second stage is more likely to perform better after inheriting the relevant scores from the first stage, which was verified in our experiment.

2 Algorithms

2.1 Overview

In this section, we refine the deep-classification algorithm based on the one proposed by Xue. et al[1]. For a given document, the main problem of traditional approach for classification is the large scale of categories and features, which is the main difficulty to over come in this contest. Facing this problem, a two-stage approach is to extract a subset of the whole categories which is related to the given document fast, called *Search Stage*, and, sequently, the traditional explicit method of classification could be used on the extracted categories to offer the

final category as the result, called *Classification Stage*. Meanwhile, another challenge is the deep hierarchy structure of the category tree. In our algorithm, we use the hierarchy information in the Search Stage, which improves the precision of the Search Stage.

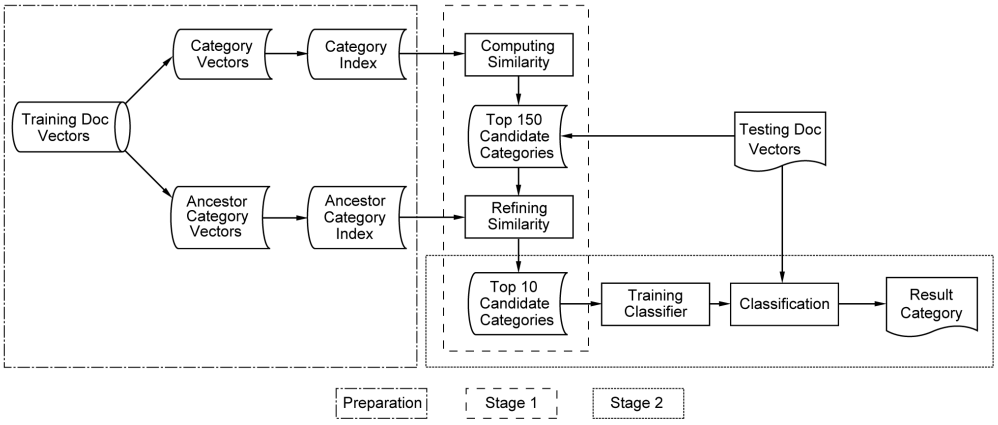

Fig. 1. Algorithm flow chart of refined deep-classification

The algorithm flow chart is shown in Fig. 1. In the *Preparation* part, we extract the category vectors by merge the document vectors belong to each category. According to the contest data, only the leaves categories of the category tree have instance of documents. Thus the ancestor categories have no document to feature them. Dealing with this case, we extract the ancestor category vectors by merge the child category vectors of them. Both the leaf vectors and the ancestor vectors are indexed in the search engine for the Search Stage.

In the Search Stage, the leaf vector index is used in the basic search part of the Search Stage, which return the top N(150, in our experiment) related categories to the given testing document; the ancestor vector index is used in the refinement part of the Search Stage, which returns the top K (10, in our experiment) related categories to the given testing document as the candidates for the Classification Stage. In our experiment, an improvement is seen after the addition ancestor information.

In the Classification Stage, the vectors of candidates are chosen to train a classifier(Naive Bayes, which works the best in our experiment). Compared with the traditional classification, this stage is based on the candidate categories from Search Stage so that for every testing case, we will train a new classifier from the candidate categories. In such case, we discover two place for improvement over the deep-classifier proposed by Xue. Firstly, for every testing case, the training data for Classification Stage is different, thus the parameters of the classifier in this stage should be different as the training data changes, which is

called *Personalized Parameter Tuning*. Secondly, as the candidate categories are recommended by the Search Stage, the ranking information of the Search Stage can be used in the Classification Stage. Both of the two improvements will be discussed in the section 2.3.

2.2 Search Stage

The main purpose of Search Stage is providing a relevant subset of category candidates for the Classification Stage. In this stage, the cosine similarity search of k-NN is chosen as the main algorithm. The input file of this stage is a set of training categories and a set of testing documents whose feature vectors were redistributed after the preparation stage and the output of this stage is top 10 categories with its relevant ranking scores. The whole stage can be separated into two parts, In the first part, a flat strategy is adopted for directly choose the top 150 leaf category candidates the its relevant ranking scores. In the second part, the ancestor information of these 150 leaf category candidates were introduced. The similarity calculation is secondly performed, but this time we calculating the similarity between the test and the candidate categories' ancestor. The final score of the these 150 candidates is a combination of scores from the first part and the second part. After seizing the final search score, an efficient top-K search is processed for extracting 10 most similar categories from 150 categories as the relevant subset of category candidates for the Classification Stage.

2.3 Classification Stage

Given the testing document and then the candidates categories and with the help of their relevance score in Search Stage, the Classification Stage builds up a traditional classifier and classify the testing document. Specifically, in our experiment, we get the best performance and efficiency using Naive Bayes Classifier to classify the 10 candidate categories.

Personalized Parameter Tuning In the standard Naive Bayes Classifier based on the multinomial model and Laplacian smoothing yields, the maximum likelihood estimation of a word ω_i and class c is:

$$p(\omega_i|c) = \frac{N_c^i + \alpha}{N_c + V\alpha} \tag{1}$$

where N_c^i stands for the number of times that ω_i occurs in the documents of c and N_c stands for the total word number of the documents of c. The total vocabulary size of the training documents is noted as V and α is the parameter of this model. In the deep-classifier, for each testing case, the Search Stage will offer a set of candidate category, of which the total vocabulary size V changes case by case. In the standard NB model, the parameter α is tuned against V to refine the performance. Therefore, in deep-classifier, as V changes against every testing case, the parameter m should also changes. We call the tuning of

the parameters of the classifier against the testing case *Personalized Parameter Tuning*.

We divide the vocabulary size in some continuous regions after the analysis of the vocabulary size distribution. For each region, the parameter α has a different value(in NB model, the larger V is, the smaller α is). If α unchanges in different regions, the scenario comes back to basic deep-classifier.

Two Stage Score Combining The K-size candidate category set from the Search Stage is not just a set in fact. As is discussed in the section 2.2, the Search Stage will return the candidate categories with the relevant score to the testing document. These scores suggest the possibility of each candidate category as the final result. After analysis of the relevant score with the final result, we discovered that the higher the score(rank) of a candidate category is, the more likely it is to be the right answer. In such case, the relevant score from the Search Stage can refine the performance of the Classification Stage.

We record the scores given by both stage and merge the two score to the final score of a candidate category. For a class c, the Equation is as below:

$$s_c = s_{1,c}^e \cdot s_{2,c} \tag{2}$$

where $s_{1,c}$ and $s_{2,c}$ stand for the score from the Search and Classification Stage respectively. The parameter e (< 1) limits the importance of the score from in the Search Stage.

Feature Selection In deep-classifier, for every test case we should train a new classifier, which takes much time. Feature selection on the candidate document feature set is significantly useful to cut down the time of the Classification Stage with little reduce of the time used in this stage.

3 Experiment and Evaluation

3.1 Experiment Specification

In this section, we describe the details of our experiment.

Category Vector Building We merge the document vectors by plus the value of the same features and combination of all the features. To get the ancestor vectors, we merge the child categories to their ancestor with the same method of merging document vectors. This process is simple enough but so useful that until now we can still not find a better method.

K-NN Similarity According to the Lucene source code [2], the tf×idf vector is not truly the value of $tf \cdot idf$. Instead, the value is $tf^{0.5} \cdot idf$. The exponent

parameter can tunes for different data set. In our experiment, we use $tf^{0.625} \cdot idf$ as the value of each vector feature.

The first part of Search Stage, we extract $N = 150$ candidates with a relevant score for each one. In the second part of Search Stage, we calculate the relevance score of the father categories of the N candidates. Then we combine the relevant score of each candidate category and their father. The final relevant score of class c in Search Stage is

$$rsc = s_c \cdot s_{f(c)}^{0.52} \tag{3}$$

where s_c stands for the first part score for c and $s_{f(c)}$ stands for the second part score for the father category of c.

Feature Selection We tried several approaches for feature selection, such as CHI-χ^2, IG, tf×idf. In our experiment, tf×idf performance the best that it doubles the speed of training with a reduce of 1% of performance on the local test data(validation.txt). However, Search Stage takes almost 12 times of time to the Classification Stage, thus the feature selection cannot improve the efficient of the whole work. In our submit code, we do not include the feature selection procession in order to make sure the best performance.

Personalized Parameter Tuning Without the feature selection, the vocabulary size for each test case is significantly different. In the test of local large data, the distribution of vocabulary size and the best value of α is given in the Table below.

vocabulary	distribution rate	tuned α
1∼2000	2.95%	0.36
2001∼3500	19.1%	0.27
3501∼6000	38.0%	0.16
6001∼10000	18.4%	0.14
10001∼ ∞	21.5%	0.08

Two Stage Score Combining Similar to the combination between the two parts of Search Stage, for class c the combination $cs_c$of two stage scores in given below:

$$cs_c = rs_c^{0.12} \cdot ns_c \tag{4}$$

where rs_c stands for the relevant score from Search Stage and ns_c stands for the score from Naive Bayes Classifier in Classification Stage. The exponent parameter can also be tuned against the data.

3.2 Experiment Conditions

Computer Hardware AMD athlon(tm) 64*2 Dual, Core Processor 5000+, 2.61GHz 7.75GB of RAM.

Operation System Microsoft Windows Server 2003 R2, Enterprise x64 Edition, Service Pack2.

3.3 Results

In section we mainly list the performance and the efficiency of the refined deep-classification on the dry-run data set.

TASK	Test Case	Hit Case	Precision	Time Used	Time per Case
Basic	1858	919	0.4946	146s	0.079s
Cheap	1858	829	0.4462	119s	0.064s
Expensive	1858	925	0.4978	158s	0.085s
Full	1858	983	0.5291	167s	0.090s

3.4 Computational Complexity Analysis

Suppose the training set have n documents covering c categories the testing set have m documents, each document have k features on an average. The complexity of the Prepare Stage is $O(nk)$. The complexity of the Search Stage is $O(cmk)+O(c) = O(cmk)$. The complexity of the Classification Stage is $O(cmk)$. So the over-all complexity is $O(cmk)$.

4 Conclusion and Future Work

In this paper, we refined the deep-classifier algorithm in both stages and get improvements on both performance and efficiency. In the future work, some parts of the algorithm will be research more deeply such as the category vector extraction from the document vector, more utilization of hierarchy information in the Classification Stage. If we can obtain the sequence of the words in the document, the N-gram method could be used, which improved the performance significantly as described in the Xue's paper [1].

References

1. Xue, G.R., Xing, D., Yang, Q., Yong, Y.: Deep Classification in Large-scale Text Hierarchies. In: Proc. of SIGIR'08, pp. 619–626. Singapore(2008)
2. Lucene Apache Website, `http://lucene.apache.org`

From Queries to Intents: a Study of Search Logs

Dingquan Wang[†⋆], Ruihua Song[†], Jian-Yun Nie[*], and Ji-Rong Wen[†]
[†] Microsoft Research Asia, [⋆] Shanghai Jiao Tong University, [*] University of Montreal
wddabc@gmail.com, {rsong,jrwen}@microsoft.com, nie@iro.umontreal.ca

ABSTRACT

Search logs are incredibly valuable to search engines because the logs contain rich information on what users want. It is important to realize users' information needs by analyzing search logs. It is also effective to enhance user satisfaction of search experiences by leveraging implicit feedback from users. Most previous works regards a query as a unique information requirement; however, we observe that users do issue various queries to look for the same target. It is far from precise if we regard a bunch of queries with the same search intent as different requirements. In this paper, we propose practical methods to model user intents and automatically group the queries with similar intents together. Evaluation results show that our proposed intent groups achieve as high precision as 0.93 and outperforms the intents based on a commercial search engine's query alterations by 36% in terms of recall. Based upon our mined intents, we conduct studies to compare intents with queries over a very large scale search logs. We find that the long tail is shortened to about 83% if distribution is based on intents rather than queries. The overlap of common intents between two weeks is larger than that of common queries by at least 5.4%. And the lifetime of body intents is dramatically increased. In addition, we show promising results of applying intents in the applications of query suggestion, ranking, and topic tracking.

1. INTRODUCTION

A search engine has become a indispensable tool for people to obtain information nowadays. As search market is competitive, search engine companies are trying all kinds of ways to improve users search experiences. Among the trials, search logs play essentially important roles. Query logs are raw resources to realize users' information needs. Caching popular queries and developing new features, such as popping up candidate queries to complete the query that a user is issuing, to target user intents is efficient and effective to enhance user experiences. Furthermore, session and click logs contain rich implicit feedback from users on what they like. For example, leveraging clicks in ranking becomes a routine and has significant contributions to ranking.

Most previous works [15] roughly regards a query as a unique information need. However, we observe that users may use hundreds of different query strings to search the same target, such as *yahoo* and *craiglist*. Diversity of queries with the same intent is common, even for a not popular search intent. For example, we find 15 different queries to look for "APA format for appendix" in three-month Bing Search's logs. Therefore, it is far from precise if we regard them as different requirements.

Only a few works pay attention to this issue. For example, query alterations (cite) are developed to help users input correct words and search relevant results. The technology is widely applied by search engines online, but it emphasizes much more on precision than recall. It is not designed for log mining offline. Clustering queries by co-clicks proves effective to improve context-aware query suggestion[8]. Inspired by their promising works, we argue that it is necessary to pay more attention to intents rather than queries in log mining. The development of identify intents technologies may help various applications based on logs.

In this paper, we propose automatically mining user intents from queries in search logs. First, we model user intents in search logs based on observations in real search logs. Second, we propose a two-layer methodology to automatically mine the queries with similar intents. Our proposed algorithms aim to identify intents with high precision and acceptable recall. Third, we create a dataset of manually labeled intents to evaluate our proposed methods and compare them with Bing Search's query alteration results. We find that the proposed query groups can obtain as high as 0.93 precision and 0.61 recall. Compared to Bing Search's query alterations, the groups improve recall by 36% by the sacrifice of 2% precision.

We conduct a series of studies on how intents influence previous log analysis based on queries over about three-month Bing Search logs. First, we estimate the scale of intents relative to that of queries. We find our intent groups reduce the scale of queries to 83%. As estimated, the number of true intents is only 50% to 60% of the number of raw queries. The huge difference between intents and queries is confirmed in our second study on the distribution of intents and the overlapped intents between adjacent weeks. If based on intents, rather than queries, the long tail is shortened by at least one fifth and the overlap between weeks increases by 5.4%. Third, the lifetime of queries is increased by intent

SIGIR'12, August, 12th-16th, 2012, Portland, USA.

groups. We find that the tail queries but corresponding to popular intents are dramatically increased in lifetime. The lifetime of body intents are most influenced, compared to those of head intents and real tail intents.

In addition, we show three applications of our mined intents by case studies and experiments. If mining intents among raw query logs, we can improve query suggestions in quality and diversity. When aggregating the clicks of all queries belonging to an intent, we can partially solve the sparsity of click-through data. A preliminary experiment proves that ranking can be significantly improved by using group clicks instead of query clicks. Also we show promising cases in which intents can help topic detection and tracking because the mined intents are closer to user's true intents than queries.

In the rest of the paper, we review related work and discuss the difference between our work and the previous work in Section 2. In Section 3, we propose two layers of heuristics for modeling intents automatically. Then, we give details on the algorithms we propose to mine intents in Section 4. We evaluate our proposed methods and compare them with the technologies of query alterations in Section 5. Also we conduct a study of large scale query logs based on intents and present some findings from Section 6 to Section 8. In Section 9, we briefly describe potential applications of our proposed intents. Finally, we draw conclusions and discuss future work in Section 10.

2. RELATED WORK

Our work mainly related to the following are: The first is query reformulation. We deploy the query log and apply several query reformulation techniques to rewrite the queries into new forms. The second is intent detection. In this part, we investigate and analysis several kinds of query features, cluster them by user intents and validate the results using user study. The third is log analysis, we change the view of the traditional query-based analysis to the intent based analysis as this paper stated.

Query reformulation is a technique that changes query terms for reducing the mismatch between the queries and documents to achieve the maximum relevance[10]. Along these techniques, spelling correction [9, 18, 2, 12, 5] is the most straightforward approach. Because the search engine is a kind of immediate editing tool, people always issue queries inadvertently. As the result, the spelling errors are more frequent in queries than the formal articles [9]. Gao et al [12] deploy the users' search log and build the n-gram language model in the source channel model [20] for correcting spelling errors effectively. Consider that our work is mainly focused on the whole query log rather than the individual queries, we can better utilize the in-log-queries' co-relation for improve grouping. The experimental results indicate that our grouping approach performs better than Gao's spell corrector. Wang et al [27] look into the query and mine the term level relevance for improving reformulation results, while the patterns are manually complied and cannot handle various types of queries as we have. Guo et al [14] incorporating the character level, term level and query level mismatches into a unified CRF model for reformulating queries. Though the result gains improvements on both precision and recall, the model is too sophisticate to process very large scale query effectively. Dang et al [10] propose to enhance reformulation relevance by applying manually anchors. This approach can reduce the query level ambiguity while has the same problems of scalability.

Intent detection, which makes the search engine smarter for the users' information needs, is gaining an increasing awareness in the Information Retrieval field. Border [6] classifies the web intents based on the topic free user targets into 3 different classes, that is *navigational*, *informational* and *transactional*. In the real search situation, however, the researchers should consider more about topic relevant intents. For example, when a user expresses the need for buying, the search engine should know not only the event that the user wants to buy, but also what's the user actually looking for . In this case, more detailed intent categories should be complied. Pu et al [23] propose two-level taxonomy with 15 major categories and 85 subcategories. Border[7] et al make the categories even more detailed . They build a commercial query taxonomy with 6000 categories by user intents.

The pre-defined classes nowadays can cover the major parts of user intents. However, using classification techniques for intent detection has two shortages: First, they cannot be updated. As the classes are predefined, a cold start problem will be faced when new intents are issued. Second, they are not personalized. As the classes are global to all users, they cannot identify the users' local information needs. One alternative way for handling the first problem is query clustering [3, 28]. Wen et al [28] utilize the cross reference between the queries and click documents to cluster queries by their intents. The solution for the second problem is applying the session log, which contains the search history of individual users, and can provide personalized profiles for different users. Cao et al [8], deploy contextual information in user's session data to predict the intents of the specific user. There are also hybrid approaches [24, 25]. Sadikov et al[24] consider a Markov model combining the document and session feature for predicting user intents. This work takes the advantages of both clustering and session, while random walk inference cannot be effectively finished when processing hundreds of millions of queries as a graph in our study . As our work mainly targets on the queries of all users rather than individuals, we apply Wen's clustering in our experiments.

Analysis the online data is an important method for the search engine to learn the real world. Corresponding to the indexing and query processing in the search technology, there are basically two types of analysis. The first is document analysis[11, 19]. Fetterly et al [11] evaluate the changes of the Web pages, where more than 150 million web pages were crawled repeatedly during 6 months and compared. This analysis on the web document tends to make the indexer smarter on recording the latest document. The second is query analysis, especially the log analysis as our work does, tends to mine the implicit pattern of the user querying behavior. Tyler et al [25] analysis the Bing search log, and generalized a individual querying pattern both related to click and session, called re-findings. In this case, the queries that users want to issue are predictable if the previous action fit this pattern. Besides the individual pattern, log analysis can also detect the group querying pattern for a large numbers of people. Google[13] once set up estimation on flu trends using the search query log, the result of which shows a strong connection between the flu activity and the distribution of flu-related query terms. Beitzel et al [4] observe the search trend variation during one day's

time and inspected this trend separately by categorizing the query topic into manually compiled classes. Compared to Beitzel's work, our work has two differences. First, the duration is longer and the data set is larger, we analysis 12 weeks of Bing search log, hence can generalize and discover some long term phenomenon. Second, we detect trends using automatic groups rather than the manually defined classes, which can provide a more comprehensive view because it is friendly to queries that from rare intents.

3. MODELING INTENTS

We present a top-down approach for understanding the user query intent, which is consisted of two levels of heuristics. The first the level is the semantic level, where queries are clustering by their clicked URLs in logs. This step is called the clustering step. The second level is the syntactic level, where the queries in the same cluster that is generated from the clustering step are grouped by their morphological and syntactical similarity, and this is the grouping step.

Before we formulate the two heuristics, three kinds of sets that partition query logs are proposed:

Definition 1. q is a query from query log, $I(q)$ is the set of queries that of the same intent with q.

Definition 2. q is a query from query log, $C(q)$ is the cluster identified by q, where:

$$C(q) = \{q' | Click(q') \cap Click(q) \neq \varnothing \wedge Doc(q') \cap Doc(q) \neq \varnothing\}$$

$Click(q)$ is the set of user clicked documents and $Doc(q)$ is the set of documents retrieved by a search engine. The equation in Definition2 is a general formulation that reveals the semantic relationship between queries by considering the clicked documents and the retrieved documents. This kind of set is identified by the clustering algorithm we choose in this study.

Definition 3. q is a query from query log, $G(q)$ is the group identifies by q, where:

$$G(q) = \{q' | q' \in C(q) \wedge Norm(q') = Norm(q)\}$$

It is routine by this definition that $G(q) \subseteq C(q)$ for each query q. $Norm(q')$ is the normalized instance of q after spelling error correction, stemming, missing order rebuilding and abbreviation mapping. This normalization phase is to discover the similarity of different queries in their string syntactic pattern. The detail of how does $Norm(q)$ is generated will be explained in Section 4.

Now we describe our heuristics based upon the above definitions:

Heuristic 1. For each query q:

$$I(q) \subseteq C(q)$$

Heuristic 2. For each query q:

$$I(q) \supseteq G(q)$$

The first heuristic reveals the positive effect for search engine features for identifying user intent. As the limitation of keywords in queries, small Levenshtein distance[26] does not mean similar user intents. For example, the queries such as *SVN* and *SVM*, while are of small distance, have a large semantic gap between each other. The former is a popular subversion control software and the latter is an algorithm in the area of machine learning.

To avoid such errors, documents retrieved or clicked are chosen as evidence to cluster queries in terms of rough search intents. Similar ideas have been applied in previous work. Wen et al[28] proposed to introduce search engine features such as click through data and retrieved document contents to enrich information for identifying similar queries. Cao et al[8] used the cross-reference between queries and the clicked URLs to building a bipartite graph and performed co-clustering algorithm, which is also what we apply in the clustering step.

The cluster boundaries defined above are insensitive of detailed intention gaps. For example, users that input "act score" or "act registration" and click the same URL `http://www.act.org` are not of the same intent even if their interest are relevant to the "act" (American College Test) stuff. Actually the clustering results are of high recall but relatively low precision. We will show supporting experimental results in Section 5. Therefore, the first heuristic only provides us an upper bound of user true intents.

The second heuristic aims to provide an approximate lower bound. We develops several rules to handle morphological and syntactical difference and discover the queries with very similar intents. Our target is high precision but maybe lower recall than the first heuristic. First, we analyze spelling errors, which are of high probabilities to occur in query logs. The identical corrected instances can be considered as sharing the same intents with high confidence. Second, the queries, which have the same query term set after a stemming process, can be also considered as of the same intent. Third, we build a contextual probabilistic model to tackle the issue of abbreviations, which often occur in query logs as we inspected. Finally, when each query is automatically normalized into a normalized label, we aggregate the queries of the same label into a group by an efficient clustering like module.

In summary, for each query q, we model the intent set $I(q)$ by inequality:

$$|G(q)| \leq |I(q)| \leq |C(q)| \quad (1)$$

Under the constrains of the two heuristics as expressed in the Inequality 1, the analysis goes straight to that the boundaries of intent $I(q)$ can be reduced by evaluating the auxiliary sets $C(q)$ and $G(q)$. Fortunately, the two sets can be automatically generated by algorithms, and thus we can conduct studies over large scale query logs.

4. MINING INTENTS

We mine clusters $C(q)$s by semantic clustering and groups $G(q)$s by syntactic grouping.

4.1 Semantic Clustering

The input of semantic clustering is click-through data. In this paper, we process the queries with at least one click, while ignoring the queries without clicks because there is no enough evidence to infer their intents.

Based on click-through data, a bipartite graph $G = (V_q, V_u, E)$ can be easily built up, where the nodes on the left side represent queries in the set V_q and the nodes on the right side represent the clicked URLs in the set V_u. We connect an edge between a query and an URL when users once queried

the query and clicked the URL. The set of edges is defined as:

$$E = \{w(v_q, v_u)|v_q \in V_q \wedge v_u \in V_u \wedge u \in Click(q)\}$$

moreover:

$$w(v_q, v_u) = \frac{freq(u|q)}{\sqrt{\sum freq(u_i|q)^2}} \quad (2)$$

$Click(q)$ means the set of URLs that are once clicked when querying q, and the $freq(u|q)$ is the number of user clicks on u after querying q. Then a co-clustering algorithm based on the bipartite graph is performed by iteratively merging the nodes and edges of small distance on both sides. The detail of this algorithm is expressed in [8].

4.2 Syntactic Grouping

The syntactic grouping module processes the queries in a cluster generated by the semantic clustering module. In this module, we make use of statistical information in the cluster and aggregate the queries based on their normalized forms. Five basic laws, which are derived from the most common synonym phenomenons that we observe in query logs, are taken into account. For each $q' \in C(q)$:

- $q' \in G(q), if\ corr(q') = corr(q)$
- $q' \in G(q), if\ stop(q') = stop(q)$
- $q' \in G(q), if\ abbr(q') = abbr(q)$
- $q' \in G(q), if\ stem(q') = stem(q)$
- $q' \in G(q), if\ vector(q') = vector(q)$

The operator *corr* corrects the query into the most frequently used case; *stop* removes stop words from the query; *stem* subtracts the terms in query into corresponding root cases; *abbr* maps some phrases in the query into their abbreviations; and *vector* represents the query by using the vector space model.

In addition, if G is a partition of cluster C, we also apply the transition law, which depicts an iterative framework of our algorithm.

> For each q_s, q_r, q_t in the same cluster, if $q_s \in G(q_r)$ and $q_r \in G(q_t)$, then $q_s \in G(q_t)$.

A straightforward solution tackling transition problem is to try all possible operations sequences on each query pair and check whether they collide the same form. However, the cost of such an approach is exponential because it has to enumerate the combination of transform action sequences. To deal with this problem, we design a graph based approach and perform the single link clustering algorithm for query grouping. In the algorithm, each query is considered as one single node. Furthermore, we define two basic graph operations:

- For each node $n \in G$, $trans(n)$ stands for updating the transformed query corresponding to n by the action $trans$
- For each node $n, n' \in G$, $coalesce(n, n')$ adds edge between n and n' if $dist(n, n') < \theta$

where, $trans(n)$ is a transform action, which is a general interface that are implemented by five instances, i.e., $corr(n)$, $stop(n)$, $stem(n)$,$abbr(n)$ and $vector(n)$. $dist(n, n')$ means the edit distance between the transformed query strings of n and n'. Please note that the transformed query of each node is updated once any transform action is applied.

In the main process of our algorithm, each node is iteratively updated by different transform actions and the order of these actions is empirically decided. After each transform action, the nodes will be checked and linked by the coalesce operation. In the last step, a large scale flooding algorithm will be performed to partition the graph into connecting sub-graphs, and the nodes in each connecting sub-graph are regarded as a query group.

4.2.1 Spelling Error Correction

Spelling errors are common in query logs. Previous research finds that there are approximately 10-15% of the queries issued to search engines contain miss-spells [9]. Thus we implement an action to correct spelling errors.

We utilize the classical noise-channel model proposed by Kernighan et al[16] as our general spell correction framework. We also incorporate the string edit distance and machine translation model proposed by Gao et al[12] as the error model in our approach. We learn local models for each cluster based on grouping its member queries when the edit distance between two queries is less than a threshold and regarding the query with maximum frequency in a group as a correction.

4.2.2 Stop words removal

We first apply the typical stop words list containing the traditional article, preposition and pronoun. Then, the term that has only one single letter is removed. Furthermore, as the particular proposition of query, the commonly-used URL terms, such as “www”, “site”, “http”, and “com”, are also considered as stop words. For example, queries “www google com” and “google” will be aggregated into the same group when the action of removing stop words is applied.

4.2.3 Abbreviation

We observe that users may use abbreviations or full names to express one entity, e.g., “American College Test” versus “ACT and “National Aeronautics and Space Administration” versus “NASA”. Abbreviations can be easy to be memorized and thus are widely used in many situations. Such pairs of abbreviations and full names are intuitively of the same intent.

In our study, an abbreviation and its corresponding full name is mined by two steps:

1) Context sensitive synonym detection. This approach is similar to the work proposed by Peng et al [21]. Each phrase is compiled into a context vector derived from the query containing the phrase. Then the query pairs with the same context vector are selected as synonym candidates for the next step.

2) Abbreviation detection by alignment. For each synonym candidate pair, the phrase of shorter length is consider as the abbreviation candidate s and the longer is considered as the full name $l =< t_1, t_2, \cdots >$, where t_i is a term of l. s is the abbreviation of l, if and only if there exists an ordered

alignment A from s to l, satisfying:

$$(\forall\ 0 < i < length(s),\ A[i]! = null) \wedge \quad (3)$$

$$(\forall\ t \in l,\ \exists\ 0 < i < length(s), A[i] = index(t_0)) \quad (4)$$

The definition of ordered alignment is same to 4.2.1. The two conditions indicate that both of the characters in s and the first character of each term in l must be aligned in A. One possible exception is that there often exists stop words in a full name and the stop words often do not appear in the abbreviation. For example "National Aeronautics and Space Administration" is abbreviated as "NASA" rather than "NAASA". That is why we set the transform action of stop words removal before the action of abbreviation.

When collecting all possible abbreviations in a cluster, we perform the longest match on the transformed queries and collapse them into their abbreviations if full names are discovered. Based on the temporary collapsed queries, the queries containing an abbreviation or its full name may be merged into one group. To reserve full text information, we choose the query containing the full name as the transformed query being on behalf of the new group.

4.2.4 Stem & Word Set Grouping

We apply the porter stemmer [22] as our general stemming algorithm. Terms are subtracted into their root cases. Then we sort the stemmed terms of each query to compose a new transformed query. For example, when applying this action, queries "bible reading online" and "read bible online " will be grouped.

Through experiments, we find that the grouping module is converged fast. Almost two iterations can produce stable results. Thus we set the number of iterations to 2 in our large scale experiments to reduce time cost.

5. EVALUATING MINED INTENTS

We conduct experiments to evaluate our proposed methods and compare the mined groups with the query alteration technology being used by commercial search engines.

5.1 Experiment Setup

To evaluate the proposed methods, we create a dataset of 100 sampled clusters, in each of which the queries with same intents are manually labeled by assessors. First, we select 100 clusters from the Bing Search logs in January 2009. The clusters contain 1,560 queries. The average number of queries in a cluster is about 16. We do not select too large clusters because it is difficult for assessors to handle. Second, we design an easy-to-use tool to assist assessors merge the queries with same intents. Queries in a cluster are shown in a list at the left and each of them has a checkbox beside. An assessor can easily check the queries with same intents and then all checked queries are shown as a merged intent group at the right. The tool also provides easy ways to modify existing intent groups. Third, assessors work on the clusters after reading the guidelines of this task and exercising on five example clusters. They can also refer to search results anytime if they are not sure about the meanings of a query. Finally, we obtain a labeled dataset, in which 589 intent groups are manually created. The average number of intents per cluster is about 6.

Given a set of queries, denoted by A, mined by an algorithm, we first align the set with the intent sets manually labeled. The intent set that has the maximum number of common queries with A is chosen as corresponding ground-truth intent set I. Then, precision and recall of A are defined as:

$$Precision(A, I) = \frac{|A \cap I|}{|A|} \quad (5)$$

$$Recall(A, I) = \frac{|A \cap I|}{|I|} \quad (6)$$

F1 measure is the geometric average of precision and recall.

Above the labeled dataset, an algorithm generate several As and we can align them with corresponding Is. To average the metrics over the algorithm result, we can apply Micro or average precision and recall as follows:

$$Micro\text{-}Precision(\{A_i\}_m, \{I_j\}_n) = \frac{\sum_{i=1}^{m} |A_i \cap I_i|}{\sum_{i=1}^{m} |A_i|} \quad (7)$$

$$Micro\text{-}Recall(\{A_i\}_m, \{I_j\}_n) = \frac{\sum_{i=1}^{m} |A_i \cap I_i|}{\sum_{i=1}^{m} |I_i|} \quad (8)$$

$$Macro\text{-}Precision(\{A_i\}_m, \{I_j\}_n) = \frac{1}{m} \sum_{i=1}^{m} \frac{|A_i \cap I_i|}{|A_i|} \quad (9)$$

$$Macro\text{-}Recall(\{A_i\}_m, \{I_j\}_n) = \frac{1}{m} \sum_{i=1}^{m} \frac{|A_i \cap I_i|}{|I_i|} \quad (10)$$

Intuitively, the Micro metrics regards each query as equal whereas the Macro metrics regards each grouped query set A_i as equal.

5.2 Evaluation Results

We evaluate the groups that our proposed methods generate and compare the groups with the method regarding each query as as an individual intent, and the semantic clusters. Results are shown in Table 1. It is not surprising that queries obtain the perfect precision and clusters obtain the perfect recall. However, the recall of queries is as low as 0.215 and the precision of clusters is 0.638 in Micro-Precision or 0.719 in Macro-Precision. Our proposed groups significantly increase the clusters in precisions. It achieves as high as 0.931 Micro-Precision and 0.978 in Macro-Precision. This indicates that our proposed method is effective to identify the queries with quite similar intents in a cluster. As our method focuses on high precisions, there is room to improve in terms of recalls.

Table 1: Evaluating different methods over manually labeled data

	Query	Group	Cluster	Bing
Micro-Precision	1	0.931	0.638	0.952
Micro-Recall	0.215	0.613	1	0.451
Micro-F1	0.354	0.739	0.779	0.612
Macro-Precision	1	0.978	0.719	0.986
Macro-Recall	0.215	0.484	1	0.390
Macro-F1	0.354	0.647	0.836	0.559

Previous works on query alterations can somehow identify the queries with similar intents. We conduct an experiment to compare our proposed method with a commercial

search engine Bing Search's alterations. In general, query alteration technologies can provide some alternatives to a query word. Some of them correct misspellings and some of them give the words with the same stem to the query word. Sometimes alternatives are wilder. For example, they could be synonyms; the alteration word could be the full name of an acronym query word or an alteration acronym is provided to be a complementary of an entity phrase in the query. We obtain all alterations of the queries in the labeled dataset from Bing Search. Despite these, we also apply stopwords filtering and term order ignored to each query, and thus the comparison would be fair. The result of Bing Search's alteration is shown in the last column of Table 1. We find that query alterations of Bing Search perform slightly better than our proposed groups in precisions, i.e., two points gap in Micro-Precision and one point gap in Macro-Precision; whereas, our proposed method performs significantly better than the query alterations in recalls, i.e., 16 points gap in Macro-Recall and 9 points in Macro-Recall.

We look closely into the generated intents and find some reasons that result in the lower recall of alterations. As the query alteration technology is applied to online queries, it is based on the knowledge that is mined from dictionaries and query logs offline. And it has to be cautious to ensure high precision because users will get angry if the alterations misunderstand their search intents. Usually the search engines leverage machine learning technologies to predict the confidence that an alteration is right for one individual query word or occasionally a query phrase and output only the alterations with enough confidence. The predictors are optimized globally. In contrary, we have semantic clusters first and we leverage the rich local information within a cluster to mine the knowledge on misspellings, acronyms, etc. Furthermore, our proposed method is applied offline and it can do a tradeoff between precision and recall. Therefore, our proposed method can identify more misspellings, in particular those of phrases, and acronyms. That is why our method outperforms query alterations in recalls.

6. ESTIMATING THE SCALE OF INTENTS

As evaluated in last section, our proposed groups obtain high precision but lower recall compared to the raw clusters. If counting the number of groups, we will over-estimate the scale of true user intents, although we are curious about the scale. It is difficult, if not impossible, to get ground-truth intents over the large scale logs of three months. However, we try to estimate the scale of ground-truth intents by leveraging the small set of labeled data used in Section 5.

The basic idea is as follows. If we have an algorithm A', the algorithm satisfies:

$$Precision(A', I) = Recall(A', I)$$

in some settings. According to the definitions of precision and recall, we can infer that $|I| = |A'|$. In other words, although both precision and recall are not necessarily equal to the perfect one, the number of wrongly misclassified items of a class happens to be the same to the number of items that should belong to the class but missing by the algorithm. Based on the fact, it is possible to estimate the number of ground-truth intents if we have such an algorithm A'.

For our addressed problem, the clustering algorithm described in Section 4.1 can be used to estimate the scale of intents. We can obtain different granularity of clusters if we change the threshold of distance[8]. Instead of clustering queries, we find that clustering groups performs better in both precision and recall if aggregating clicks of all queries in a group. Therefore, we conduct a series of clustering experiments based on groups over the 100 clustered that are labeled by assessors. To average precision and recall of 100 clusters, we apply two frequently-used methodologies: Macro average and Micro average. As a results, we get four curves as shown in Figure 1. The threshold is changed from 0 to 2. As shown in the figure, when the threshold is set as 0.6, Micro-Precision and Micro-Recall cross; whereas, Macro-Precision and Macro-Recall cross when the threshold is 1.25.

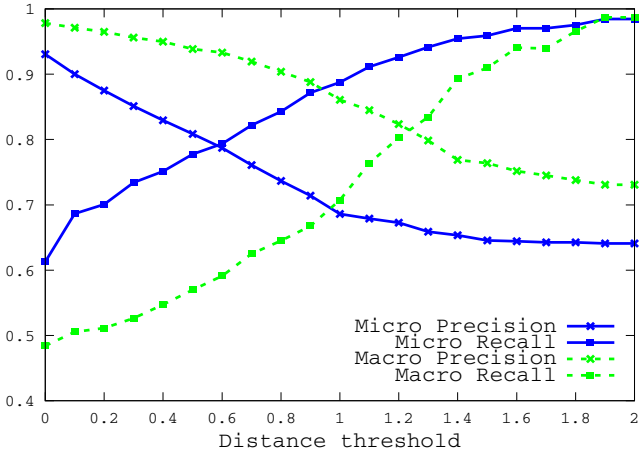

Figure 1: Distance threshold on precision&recall

As Micro average regards all queries equally while Macro average regards all clusters equally, we apply the two thresholds to estimate the scale of true intents. First, we cluster all queries in the 12-week search logs and group similar queries in each cluster. Then, we aggregate all clicks of a group and re-cluster the groups by applying the threshold 0.6 and 1.25 respectively. Finally, we count the number of queries, groups, estimated intents, and raw clusters. Results are shown in Table 2.

Table 2: Scale of 12-week query logs based on different granularity

Granularity	Scale	Ratio
Query	506,851,706	-
Group	420,971,203	83.1%
Re-clustered Intent (thr=0.6)	299,593,161	59.1%
Re-clustered Intent (thr=1.25)	250,590,314	49.4%
Cluster	231,144,764	45.6%

We observe that the scale of intents is not as large as the number of unique queries. Based on groups with high precision, the number of groups decreases to about 83% of the number of queries. Based on clusters with high recall, the number of clusters is only 46% of that of queries. The number of user intents would be between the two numbers. As we estimate, the number of user intents is probably between about 50% and 60% of the number of queries. It is somehow surprising. Although we have such many queries, the amount of real user intents is only half of the number.

7. RE-EXAMINING LOGS BASED ON INTENTS

Based upon our mined groups and clusters, we re-examine two basic properties of the 12-week query logs.

7.1 Long Tail Distribution

We plot query frequencies in a descending order in Figure 2. The axis could stand for queries, groups, or clusters. The query frequency of a group is defined as the sum of frequencies of all queries belonging to the group. We call the query frequency of a group as group frequency for short. Cluster frequency is defined in a similar way. To be simple, query frequency refers to the frequency based on queries in the remaining parts.

Previous works discovered that frequencies based on queries are in a Zipf distribution. As shown in Figure 2(a), a long tail corresponds to the huge amount of queries that have been queries only once in the 12 weeks. When we group queries with similar intents together, the distribution of group frequency is also Zipf (See Figure 2(a)); however, more queries move close to y axis with their groups. As a result, the long tail is not so long as that of query frequency. This indicates that the intents behind some long tail queries are not that rare. They may be variants of a head or body queries. Similar to group frequency, cluster frequency distribution has an even shorter tail as shown in Figure 2(c). It is easy to understand because more related queries are put into less clusters.

7.2 Overlap between Weeks

Another interesting question about search logs is how stable users' interests are. Sometimes a query is interested this week but will never be asked next week. Such a query does not represent a user's stable interest. Different the unit of time, such as a week and a month, can be used. We choose a week as our time unit as we conduct the study on the 12-week search logs. We can do statistics on the overlap between two adjacent weeks and get a trustable average overlap.

Based on queries, we have a set of queries issued in every week. Then we calculate Jaccard coefficient between any two sets of adjacent weeks. Jaccard coefficient is defined as the size of the intersection divided by the size of the union of the sample sets. When a set contains unique queries only, we multiply a query's frequency when calculating Jaccard coefficient. For example, the Jaccard coefficient between week 1 and week 2 is as follows:

$$J(Q_{w1}, Q_{w2}) = \frac{\sum_{q_i \in Q_{w1} \cap Q_{w2}} (f_{q_i,w1} + f_{q_i,w2})}{\sum_{q_j \in Q_{w1} \cup Q_{w2}} (f_{q_j,w1} + f_{q_j,w2})}$$

If based on groups, we calculate Jaccard coefficient in a similar way:

$$J(G_{w1}, G_{w2}) = \frac{\sum_{g_k \in G_{w1} \cap G_{w2}} \sum_{q_i \in g_k} (f_{q_i,w1} + f_{q_i,w2})}{\sum_{g_l \in G_{w1} \cup G_{w2}} \sum_{q_j \in g_l} (f_{q_j,w1} + f_{q_j,w2})}$$

Here, $f_{q_i,w1}$ and $f_{q_i,w2}$ are the frequency of query q_i in week 1 and week 2. If the query does not occur in a week, the frequency is zero. Given two weeks, the denominators of coefficients based on queries or groups are same and thus the overlap between weeks is comparable. The Jaccard coefficient based on clusters is defined in the same way to that based on groups.

For each granularity, we average the 11 coefficients of adjacent weeks and obtain the average overlap between weeks. The results are shown in Table 3. First, we observe that about 68.9% are common interests between adjacent weeks even based on queries. This confirms previous observations on users often issue the same queries over time. Second, when the queries with similar intents are grouped, the average overlap between weeks increases by about 5.4%. It indicates that sometimes users issue slightly different queries for similar information needs. Thus the overlap based on groups is obviously increased. Third, the average overlap based on clusters is as high as 80%. This somehow gives an upper-bound of truly overlapped intents. As our proposed groups do well in high precision rather than high recall, we believe that the amount of users' interests that are stable through weeks is between 72.7% and 80%

Table 3: Average overlap between two adjacent weeks over the 12-week query logs based on different granularity

Granularity	Average Overlap	Increase(%)
Query	0.689	-
Group	0.727	5.4
Cluster	0.800	16.2

8. INVESTIGATING THE LIFETIME OF INTENTS

In this paper, we define the lifetime a query as the number of days when the query occurs during the given period, here the first 12 weeks in 2010. Accordingly, we define the lifetime of an intent, including a group or a cluster, as the number of days in the union of the days when its member queries occur. For example, given a group g, which has two member queries q_1 and q_2, if q_1 occurs in days d_1, d_2 and q_2 occurs in days d_2, d_3,

$$Lifetime(q_1) = |\{d_1, d_2\}| = 2;$$

$$Lifetime(q_2) = |\{d_2, d_3\}| = 2;$$

and

$$Lifetime(g) = |\{d_1, d_2\} \cup \{d_2, d_3\}| = 3.$$

We expect that the lifetime of intents would be extended compared to that of individual queries. In this section, we investigate the changes on lifetime over the 12-week search logs and our mined intent groups.

8.1 Query vs. Group

We investigate the increase ratio of lifetime from a query to its group. The increase ratio is defined as:

$$IncreaseRatio(q, g(q)) = \frac{Lifetime(g(q)) - Lifetime(q)}{Lifetime(q)}$$

where $g(q)$ is the group that q belongs to.

We randomly sample some queries and retrieve their belonging groups. Then, we get the lifetime of the sampled queries and their groups and calculate the increase ratio of lifetime for each query. Next, we sort all queries by query frequency in a descending order. When queries are equal in terms of frequency, we sort queries by the increase ratios of

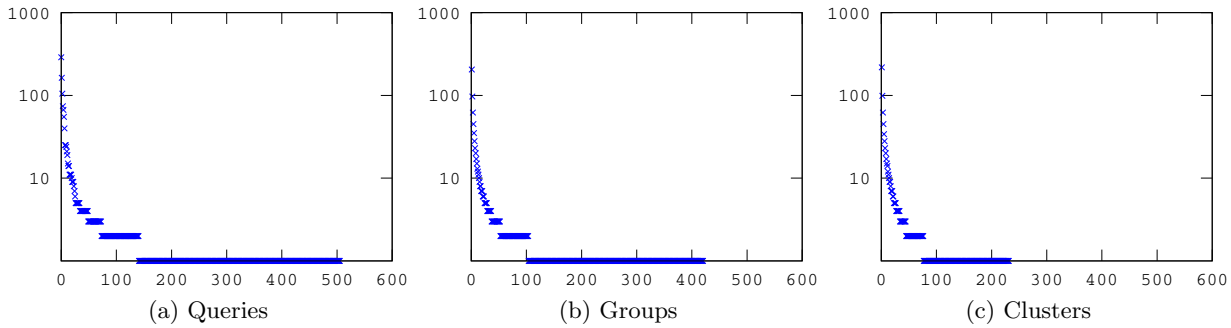

Figure 2: Frequency distribution based on different granularity

lifetime also in a descending order and called the maximum as the largest increase ratio. Finally, the results are shown in Figure 3.

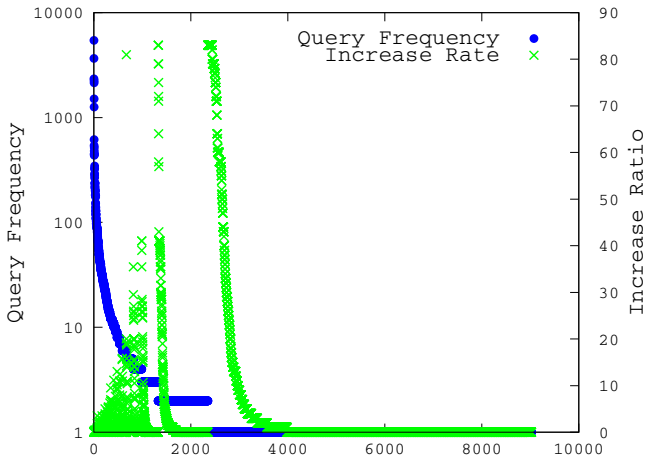

Figure 3: Comparing Increase Ratio and Query Frequency

In the figure, we find an increasing slope of the largest increase ratios of different query frequencies as the frequencies go down. Some tail queries, such as googld (it is actually a misspelling of google), are rarely queried, e.g., googld's lifetime is only 1, but the real intents behind them are popular, e.g., google's lifetime is 84. As a result, the lifetime increase ratio of googld reaches 83, i.e. the maximum over the 12-week data. On the contrary, some head queries, such as google, have already occurred almost everyday, and thus the increase ratios are limited. For each set of queries with the same frequencies, we also observe a power-like distribution of increase ratios, as shown in Figure 3. This indicates that there are considerable amount of popular intents corresponding to the rare queries.

In addition, the increase ratios of lots of queries that are asked only once is 1, which means that there is no change even after intent grouping. These kinds of queries are truly tail queries with infrequent intents. We look closely at the queries and find that most of these queries carry some very specific information needs(called informational queries[6]). Different from the head queries and the rare queries with large increase ratio, these true rare queries typically have more words in the queries. The head queries and the rare queries of high increase ratio are often typed for visiting a specific websites(called navigational queries), which have comparatively less words than informational queries.

8.2 Group vs. Group Leader

We investigate how intent grouping influences the lifetime of group leaders. A group leader is defined as the most frequently-asked query in a group. Usually the group leader also has the longest lifetime. In this experiment, we first randomly sample some groups and retrieve their group leaders. Then we calculate the lifetime of each group and its group leader. We plot each group in Figure 4 and the groups are ordered in terms of their lifetimes in a descending order. We also plot their corresponding group leaders in the same position as the groups are and the y-values are the lifetimes of group leaders.

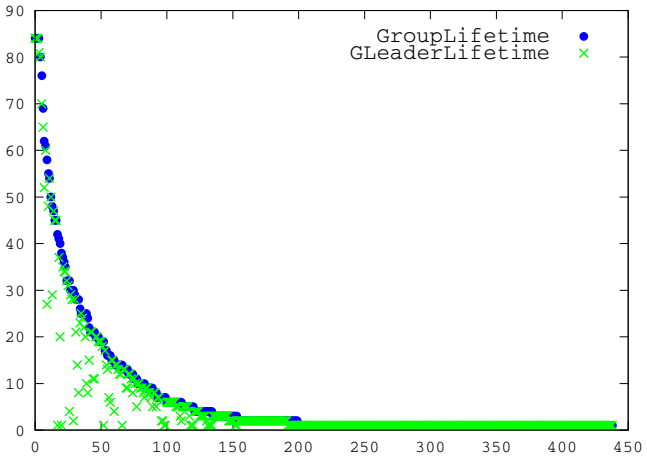

Figure 4: Gap of lifetime between a group and its group leader

According to Figure 4, we find that the grouping process has more influence on the "body" queries rather than the head or tail queries. Some most popular queries, such as *craiglist*, *facebook*, etc., are queried everyday, and thus their lifetimes are no longer extended because of grouping. Some less popular queries, such as *figure skating schedules*, are not queried everyday, but if counting all similar queries with same intents together, the lifetime is dramatically extended. Some real tail queries, such as *mini bus service boston to burlington*, cannot be grouped with others, and thus the lifetime stays short.

9. APPLICATIONS

Our mined intents can be used in many applications based on query mining, such as query suggestion, query mining for improving ranking, topic detection and tracking of query logs, etc.

9.1 Query Suggestion

Previous query suggestion technologies have done a lot of work to filter low-quality queries to avoid suggest bad queries to users. Our proposed intent mining methods provide a new way to group queries with similar intents. By selecting the best one in a group, we can easily filter many low-quality queries, including misspellings and rare representation of the intents, and duplicated queries although they are different if we only reply on string comparison algorithms. For one instance, without applying our proposed methods, the suggestions for query "nyc parking calendar" are as follows:

(1) nyc parking rules 19
(2) alternate side parking 15
(3) nyc alternate side parking 12
(4) alternate side parking new york city 6
(5) alternative parking nyc 3
(6) nyc parking regulations 3
(7) 2011 holiday dot calendar 3
(8) alternate side nyc 3
(9) nyc alternate side calendar 3
(10) new york alternate parking 2
(11) parking rules in nyc 2
(12) parking regulations nyc 2
(13) alternate side parking in new york 2
(14) alternate side parking nyc 1
(15) althernative parking in nyc 1
(16) ny alterbnte parking 1

When we group similar queries, we suggest the following queries:

nyc parking rules 21 (1)(11)
alternate side parking 15 (2)
nyc alternate side parking 21 (3)(4)(13)(14)
2011 holiday dot calendar 3 (7)
alternative parking nyc 7 (5)(10)(15)(16)
nyc parking regulations 5 (6)(12)
alternate side nyc 3 (8)
nyc alternate side calendar 3 (9)

Apparently, the results based on our proposed intents are more informative and better in quality than the results based on queries.

For another instance, the suggestions for query "justin bieber concerts" are as follows:

(1) justin bieber concert tickets 32
(2) justin bieber tour dates 23
(3) justin beiber concert 17
(4) justin bieber concert 13
(5) justin bieber on tour 11
(6) justin bieber concert dates 8
(7) justin bieber concert schedule 8
(8) justin beiber concert schedule 8
(9) justin bieber tour schedule 6
(10) justin beiber concerts 6
(11) justin beiber concert dates 5
(12) justin bieber in concert 4

When we group similar queries, we suggest the following queries:

justin bieber concert tickets (1)
justin bieber tour dates (2)
justin bieber on tour (5)
justin bieber concert dates (6)(11)
justin bieber concert schedule (7)(8)
justin bieber tour schedule (9)

Some near-duplicated suggestions are removed from the results based on our proposed intents, e.g.,

justin bieber concerts (3)(4)(10)(12)

9.2 Query Mining for Improving Ranking

Previous works show that click-through data are valuable and useful in improving ranking of Web documents in Web information retrieval. However, it is a known problem that click-through data have sparse issues, in particular for tail queries. When a query has been searched only several times, it is difficult to accumulate enough clicks as evidence of re-ranking documents. As a result, this kind of queries cannot be improved by click-through data as some popular queries are.

Our proposed intent mining methods can partially address the problem. As Experiment xx indicates, some search intents are not as rare as they are if we group the queries with similar intents together. That's to say, although a rare query is searched only a few times, some others may query the same intent by using other queries and they clicked. If we aggregate all clicks for each intent group, we can expect that clicks increase. Thus every query in the group can share the clicks. In such a way, the click-through data are no longer as sparse as those based on unique queries.

In this paper, we use a large-scale query set that contains 13,920 queries. On average, there are more than 30 documents are judged in five-grade relevance levels. When we intersect the query set with our processed 12-week query logs, there are 8,357 common queries, 7,453 of which are grouped with some other queries in our proposed intent groups. We conduct experiments upon the 7,453 queries to verify whether intents have potentials to improve ranking by enhancing click-through data.

We do statistics of both query based click-through data and group based click-through data for the 7,543 queries. We count the number of unique URLs that have been clicked and the number of clicks. Results are shown in Table 4. It indicates that our proposed intent groups can increase the number of clicked URLs by 263% and the number of clicks by 351% over query based click-through. These numbers confirm our expectation that intent groups can alleviate the sparse issue of click-through data.

Table 4: Compare the amount of click-through data based on queries and groups

	#urls	**#clicks**
Group	1,058,669	708,918,600
Query	291,854	157,357,537
Group/Query	3.63	4.51
(Group-Query)/Query	263%	351%

There are different methods to take use of click-through

data to improve ranking, e.g. [1, 15]. As how to use click-through data is not our focus, we just apply a na?ve method to compare two kinds of click-through data, i.e. query clicks and group clicks. First, we calculate content based relevance scores by the BM25 formula for all retrieved documents. Second, we use click-through data to re-rank the top 20 documents in the first step. If query clicks are applied, we sort the top 20 documents for each query by the number of clicks based on queries in descending order. If there is no corresponding click, the number of click is set as zero. If two documents have the same number of clicks, they are ordered by their BM25 scores. Similarly, if group clicks are applied, we re-ranked the top 20 documents for each query by the number of clicks based on groups. Third, we evaluate the three methods by NDCG measures. The results are shown in Table 5.

As Table 5 indicates, both query clicks and group clicks significantly improve the content based baseline method in NDCG measures. Their improvements over the baseline method are larger than 9 points in terms of NDCG@1, and the improvements are larger than 2.7 points in terms of NDCG@5. Such significant improvements confirmed that click-through data are an important source to enhance Web search as the data contains users' valuable implicit feedback. It is significant to improve click-through data for Web search.

When comparing the two click-based methods, we find that the group click based method improves the query click based method by 0.18 points in terms of NDCG@5 and the improvement is statistically significant. This indicates that our proposed intent groups do help enhance the quality of click-through data. In addition, group clicks affect 710 more queries than query clicks. That's to say, the influence of group based click-through increases that of query based click-through by 12%. This indicates that our proposed intent groups also improves click-through data in coverage.

9.3 Topic Detection and Tracking

Query logs are useful to detect interested topics and track their trends. Most previous work is based on query strings[17, 29]. Our proposed intents can group queries with similar intents together. If applying topic detection and tracking technologies based on the intents, we argue that results will be more precise in counting the coverage of a topic. For example, we trace one event that happened during the period our target query log comes from. This event is the "Grammy Awards ceremony " that promoted at the beginning of every year. To highlight the difference of our approach, we track the query log based on 3 points of views. The first is query view, in which we simulate the former detecting approach by term matching. The second is group view, we firstly used the query terms to retrieve the group they belong to. Then we replace the query by its group. The group frequency is the sum of all the queries' frequency it contains. The third is cluster view, the method is same as the group approach. We track the query "Grammy" for 12 weeks in our query log, and record the trends as shown in Figure 5

Have changed the view, we find that the event is much more popular than the traditional statistics. The reason for this finding is that even we have taken all kinds of relevant term into account, we still can not simulate the user behavior. Because they tends to issue some synonym and error words for reflecting their popular intents. As the result, the traditional term matching approach will miss these cases.

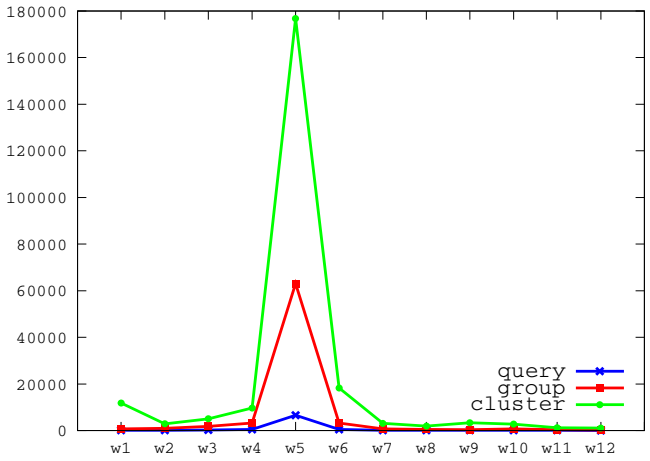

Figure 5: Frequency trend of query "Grammy"

Actually when grouping queries with similar intents, we observe that the number of related queries in No.1 week to No.10 week dramatically increases to about 6 10 times. The peak occurs at No.5 week when Grammy Awards ceremony was held.

10. CONCLUSION AND FUTURE WORK

11. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26. ACM, 2006.

[2] F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In *Proceedings of the 5th conference on HLT and EMNLP*, pages 955–962. ACL, 2005.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the 6th international ACM SIGKDD conference on Knowledge discovery and data mining*, pages 407–416. ACM, 2000.

[4] S. Beitzel, E. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 321–328. ACM, 2004.

[5] E. Brill and R. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting on ACL*, pages 286–293. ACL, 2000.

[6] A. Broder. A taxonomy of web search. In *ACM SIGIR forum*, volume 36, pages 3–10. ACM, 2002.

[7] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 231–238. ACM, 2007.

Table 5: Comparing retrieval effectiveness of the baseline method and two click based methods. Notes: "*" means a method is statistically significantly better than the baseline BM25 method. "" means a method is statistically significantly better than both the baseline method and the other compared method. We apply t-test in significant testing and significant difference is concluded when p-value is less than 0.05.**

	NDCG@5	Improvement	#QAffected	NDCG@1	Improvement	#QAffected
BM25	55.88	-	-	56.78	-	-
Query Clicks	58.65	2.77*	5913	65.78	9*	2624
Group Clicks	58.83	2.95**	6623	65.95	9.17*	2595

[8] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceeding of the 14th international ACM SIGKDD conference on Knowledge discovery and data mining*, pages 875–883. ACM, 2008.

[9] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP*, volume 4, pages 293–300, 2004.

[10] V. Dang and B. Croft. Query reformulation using anchor text. In *Proceedings of the 3rd international ACM WSDM conference*, pages 41–50. ACM, 2010.

[11] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of web pages. *Software: Practice and Experience*, 34(2):213–237, 2004.

[12] J. Gao, X. Li, D. Micol, C. Quirk, and X. Sun. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd international COLING conference*, pages 358–366. ACL, 2010.

[13] J. Ginsberg, M. Mohebbi, R. Patel, L. Brammer, M. Smolinski, L. Brilliant, et al. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–4, 2009.

[14] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 379–386. ACM, 2008.

[15] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161. ACM, 2005.

[16] M. Kernighan, K. Church, and W. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th international COLING conference*, volume 2, pages 205–210. Association for Computational Linguistics, 1990.

[17] A. Kulkarni, J. Teevan, K. Svore, and S. Dumais. Understanding temporal query dynamics. In *Proceedings of the 4th international ACM WSDM conference*, pages 167–176. ACM, 2011.

[18] M. Li, Y. Zhang, M. Zhu, and M. Zhou. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st international COLING conference and the 44th annual meeting of the ACL*, pages 1025–1032. ACL, 2006.

[19] A. Ntoulas, J. Cho, and C. Olston. What's new on the web? the evolution of the web from a search engine perspective. In *Proceedings of the 13th international WWW conference*, pages 1–12. ACM, 2004.

[20] F. Och. Statistical machine translation: from single-word models to alignment templates. *Germany, RW TH Aachen*, 2002.

[21] F. Peng, N. Ahmed, X. Li, and Y. Lu. Context sensitive stemming for web search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 639–646. ACM, 2007.

[22] M. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[23] H. Pu, S. Chuang, and C. Yang. Subject categorization of query terms for exploring web users' search interests. *Journal of the American Society for Information Science and Technology*, 53(8):617–630, 2002.

[24] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *Proceedings of the 19th international WWW conference*, pages 841–850. ACM, 2010.

[25] S. Tyler and J. Teevan. Large scale query log analysis of re-finding. In *Proceedings of the 3rd international ACM WSDM conference*, pages 191–200. ACM, 2010.

[26] R. A. Wagner and M. J.Fischer. The string-to-string correction problem. *Joural of the Association for Computing Machinery*, 1974.

[27] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceeding of the 17th ACM CIKM*, pages 479–488. ACM, 2008.

[28] J. Wen, J. Nie, and H. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1):59–81, 2002.

[29] E. Yom-Tov and F. Diaz. Out of sight, not out of mind: on the effect of social and physical detachment on information need. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, pages 385–394. ACM, 2011.

Advertising Keywords Recommendation for Short-text Web Pages using Wikipedia

WEINAN ZHANG, DINGQUAN WANG,, Shanghai Jiao Tong University
GUI-RONG XUE, Aliyun.com
HONGYUAN ZHA, Georgia Institute of Technology

Advertising keywords recommendation is an indispensable component for on-line advertising with the keywords selected from the target Web pages used for contextual advertising or sponsored search. Several ranking-based algorithms have been proposed for recommending advertising keywords. However, for most of them performance is still lacking, especially when dealing with short-text target Web pages, i.e., those containing insufficient textual information for ranking. In some cases, short-text Web pages may not even contain enough keywords for selection. A natural alternative is then to recommend relevant keywords not present in the target Web pages. In this paper, we propose a novel algorithm for advertising keywords recommendation for short-text Web pages by leveraging the contents of Wikipedia, a user-contributed on-line encyclopedia. Wikipedia contains numerous entities with related entities on a topic linked to each other. Given a target Web page, we propose to use a content-biased PageRank on the Wikipedia graph to rank the related entities. Furthermore, in order to recommend high quality advertising keywords, we also add a advertisement-biased factor into our model. With these two biases, advertising keywords that are both relevant to a target Web page and valuable for advertising are recommended. In our experiments, several state-of-the-art approaches for keyword recommendation are compared. The experimental results demonstrate that our proposed approach produces substantial improvement in the precision of the top 20 recommended keywords on short-text Web pages over existing approaches.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Search process

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Contextual Advertising, Advertising Keywords Recommendation, Topic-sensitive PageRank, Wikipedia

1. INTRODUCTION

In the last decade, on-line advertising has become a prominent economic force and the main income source for a variety of Web sites and services. According to Interactive Advertising Bureau's (IAB[1]) annual Internet advertising report, yearly advertising revenues have grown from US$6.01 billion in 2002 to US$26.04 billion in 2010 [IAB and PricewaterhouseCoopers 2011]. Some reports have already shown that on-line advertising has overtaken TV to become the largest advertising medium in countries such as UK [Sweney 2009]. Among the several existing on-line advertising channels, search advertising, a search engine based method of placing ads on Web pages, has

[1]Interactive Advertising Bureau. http://www.iab.net

Author's address: Weinan Zhang and Dingquan Wang, Dept. of Computer Science and Engineering, Shanghai Jiao Tong University, No. 800, Dongchuan Road, Shanghai, China 200240; Gui-Rong Xue, Senior Director, Aliyun.com, Zhejiang, China; Hongyuan Zha, College of Computing, Georgia Institute of Technology, Atlanta, GA 30032.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

become the driving force behind the large-scale monetization process of Web services through on-line marketing [Cristo et al. 2006].

Advertising keywords recommendation is an indispensable process of *sponsored search* and *contextual advertising*, which are the two main approaches of search advertising. In sponsored search, the ads are displayed at the top or the right of the result pages of search engines, largely based on the degree of matching between the keywords of user queries and advertiser's bid keywords. The clicks on these ad links will take users to the Web pages of advertisers, generally called landing pages. A natural question for an advertiser is which keywords should be bid on for her landing pages? As users can express their search intent in a variety of different queries, it is almost impossible to conceive all the relevant keywords for the landing pages [Cristo et al. 2006]. Thus sponsored search system, as an important service for the advertisers, provides recommendations of advertising keywords for the landing pages. In contextual advertising, on the other hand, it is also desirable to display relevant ads on the target Web pages [Anagnostopoulos et al. 2007]. This is mostly done by first extracting advertising keywords from the target Web pages and then retrieving the relevant ads using those advertising keywords. Figure 1 illustrates the keyword-based contextual advertising process. In summary, it is essential to accurately extract advertising keywords in order to display highly relevant ads, both in sponsored search and in contextual advertising.

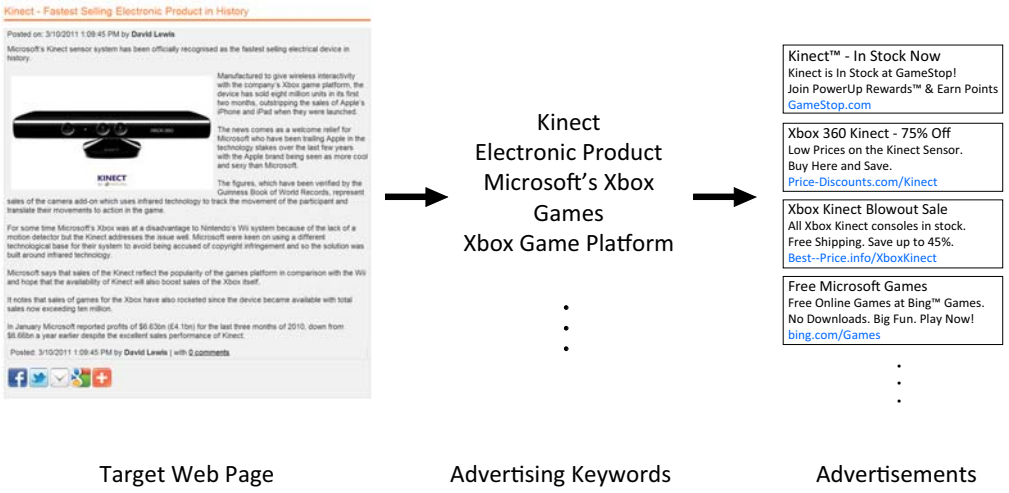

Fig. 1. An example of keyword based contextual advertising process.

Because of its importance, it is not surprising that a variety of approaches for advertising keywords recommendation for Web pages have been proposed including the supervised learning based algorithm proposed by Yih et al. [Yih et al. 2006], the KEA system [Fang et al. 2005; Jones and Paynter 2001; Witten et al. 1999], and the unsupervised learning algorithm proposed by Matsuo [Matsuo 2003]. However, those existing advertising keywords recommendation algorithms largely rely on the textual content of a Web page itself despite of the fact that a substantial number of Web pages mainly contain multimedia contents such as images or videos. As an illustration, we collected statistics about the distribution of word count of the Web pages from Open Directory

Project[2] in Figure 2. From Figure 2, we find that 27.24% pages in ODP has less than 100 words, which is generally called *short-text Web pages* in this paper. If we consider ODP as a reasonable representation of an essential part of the whole Web, we can conclude that a substantial proportion of pages on the Web contain very little textual contents.

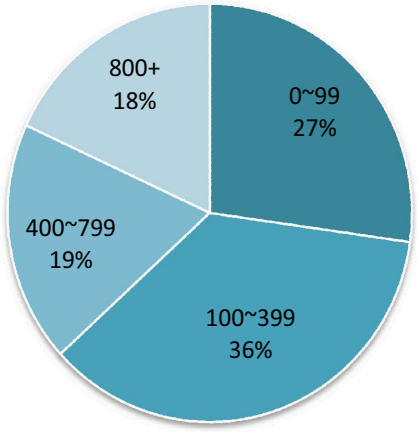

Fig. 2. Distribution of the word size of the Web pages in ODP

It is not surprising traditional advertising keywords recommendation algorithms do not work well on these short-text Web pages. We can single out two main reasons: Firstly, the short-text Web pages offer less textual information. They probably contain very simple content structure and the content is poor, which makes it difficult for recommendation system to rank the keywords well and thus leads to low accuracy. Secondly, in some cases, the situation is even worse, the short-text Web pages do not contain enough candidate terms or phrases.

A natural idea to overcome above two issues is to enrich the set of advertising keywords by introducing new advertising keywords which do not occur in, but are still relevant to, the target Web pages. There are two possibilities. The first is to simply enrich the content of target Web pages and then use the enriched content to do the keywords recommendation work [Ribeiro-Neto et al. 2005]. The second one, which we develop in this paper, requires to analyze the relationship between advertising keywords and then obtain new advertising keywords that are semantically relevant to the existing ones. So how can we identify the relationship between keywords? It turns out that Wikipedia[3] is an ideal resource to do that: It is a Web based collaborative encyclopedia and contains, for example, more than 3 million entities in English language. The entity articles cover a diverse set of topics in a large number of areas. And the number of its entities is still growing rapidly. Moreover, each entity is described by a relatively complete and concise article with hyperlinks linking to other Wikipedia entities indicating the semantic relationship between them. Therefore, they can be exploited to obtain high quality advertising keywords relationship for recommendation.

[2]Open Directory Project. http://www.dmoz.org
[3]Wikipedia. http://www.wikipedia.org

In this paper, we propose a novel approach of advertising keywords recommendation that makes use of entities and links from Wikipedia. As mentioned above, our approach can recommend advertising keywords that are highly relevant to the target Web page even if they do not occur in it. These keywords are called *leveraged keywords* in this paper. Our approach makes use of Wikipedia entites as the dictionary to recommend keywords. The usefulness of Wikipedia entities is evidenced by the fact that more than 99.8% of ODP Web pages contain one or more Wikipedia entities[4]. This high proportion makes Wikipedia a valuable thesaurus for the Web pages.

Structurally, Wikipedia can also be viewed as a directed graph with vertices and edges corresponding to its entities and links among the entities, respectively. This allow us to generate the related advertising keywords by propagating the keywords on the graph using Markov Random Walk. Specifically, we will use the algorithm of PageRank to implement the propagation process. Furthermore, inspired by topic-sensitive PageRank proposed in [Haveliwala 2002], we introduce two kinds of bias, namely *content bias* and *advertisement bias*, into the propagation process, making it possible to recommend advertising keywords that are both relevant to a target Web page and valuable for advertising. In our experiments, we compare our algorithm to several baseline and state-of-the-art algorithms for advertising keywords recommendation. In particular, we focus on evaluating our algorithm on short-text Web pages. The result shows that our approach achieves substantial improvement over the supervised learning approach, measured by the precision of the top 20 recommended keywords, demonstrating the effectiveness of our algorithm.

The main contributions of our work are summarized as below.

— The problem of the keywords recommendation for short-text Web pages is emphasized.
— A two-stage approach is proposed to solve the problem. In the first stage, candidate keywords are extracted from the target Web page while in the second stage, related keywords to the target Web page are recommended, using a random walk based algorithm applied to the Wikipedia graph. The experimental result shows a significant improvement on the recommendation performance for short-text Web pages.
— To the best of our knowledge, our proposed content & advertisement-sensitive PageRank is the first of its kind for multi-topic-sensitive PageRank algorithms.

The rest of this paper is organized as follows. In section 2, we discuss several related work about search advertising, keywords recommendation, and application of Wikipedia. Some preliminary work is presented in section 3. In section 4, we present our approach using the content & advertisement-sensitive PageRank on Wikipedia graph. In section 5, we describe the experiments and analyze the results. Finally, we present our conclusion and future work in section 6.

2. RELATED WORK

2.1. Search Advertising

According to the Internet Advertising Revenue Report to the year 2010 [IAB and PricewaterhouseCoopers 2011], 45% of the total revenue from online advertising in United States is contributed by search advertising, which continues to lead in the market and is followed by Display Banners (26%) and Classifieds (9%). Searching advertising is a search engine based approach of placing on-line ads on Web pages. It is an interesting subfield of Information Retrieval that involves large scale search, content analysis, information extraction, statistical models, machine learning, and microeconomics. De-

[4]This statistic work is made by us.

scribed by IAB, the two main forms of search advertising are *sponsored search* and *contextual advertising*.

2.1.1. Sponsored Search. When a query is submitted to the search engine, two searches are performed. The first one is *organic search* which returns the Web pages with relevant content. The second one is sponsored search which returns the paid ads [Becker et al. 2009]. Sponsored search was introduced by Overture[5] in 1998 and now Google[6] offers the largest service. The search engine retrieves the ads of sponsors mainly by the keywords of the user query and displays them on the top or right of the search result pages. Generally, there are three forms of cost: *Cost-Per-Click* (CPC), *Cost-Per-Mille* (CPM), and *Cost-Per-Action* (CPA). The sponsored search system makes auctions on every keyword and the advertisers bid on some keywords for their ads. It is more likely for their ads to be ranked higher if the advertisers pay more for the impressions and clicks of their ads.

The ranking mechanism for sponsored search decides which ads retrieved should be ranked higher. In the work of [Feng et al. 2003], two mainstream ranking mechanisms are compared: *ranking by willingness to pay* (WTP) and *ranking by willingness to pay* $\times$ *relevance* (WTP $\times$ Rel). Through computational simulations, they found WTP $\times$ Rel performs better in almost all cases, while WTP is better when the correlation between the relevance and WTP is large.

Besides the works on ranking mechanism, more academic researches focus on matching strategy for the improvement of the relevance between ads and user queries [Hillard et al. 2010; Broder et al. 2008; Raghavan and Hillard 2009]. Since the content of ads and user queries are both short, short content matching algorithms are used. Some query expansion based work is presented in Section 2.2.2. Other work makes use of some external information such as the content and types of landing pages [Choi et al. 2010; Becker et al. 2009].

2.1.2. Contextual Advertising. Contextual advertising, introduced by Google[7] in 2002, refers to the placement of ads on the third party Web pages based on the content of the target Web pages and the ads. The publishers and search engine will share some revenue once any ad on their Web pages is clicked. Some studies [Wang et al. 2002] have already shown that the relevance between the content of target Web pages and the ads makes a large difference at the click-through rate. Intuitively, the content of target Web pages suggests the users' interest and if the ads are relevant to the Web page content, they are more likely to attract the users. Therefore, the matching work of the target Web pages and the ads is the key point of contextual advertising.

Keyword based approaches are widely used in contextual advertising. This kind of approaches first extract keywords from the target Web pages and then use these keywords to retrieve the ads just like sponsored search. However, due to the vagary of keywords extraction and the lack of Web page content, the keyword based approaches always lead to irrelevant ads. The work of [Ribeiro-Neto et al. 2005] is a typical keyword based approach, which matches the ads with the target Web pages content and extracted keywords to get the winning strategy. Besides keyword based approaches, the authors in [Broder et al. 2007] make use of semantic information to enhance the matching work. They classify both pages and ads into a common taxonomy and merge the keywords matching work with the taxonomy matching work together to rank the ads. As analyzing the entire page content is costly and thus new or dynamically created Web pages could not be processed to match the ads ahead of time, the authors in

[5]Overture. http://www.overture.com
[6]Google Adwords. http://adwords.google.com
[7]Google AdSense. http://www.google.com/adsense

[Anagnostopoulos et al. 2007] proposed a summarization based approach to enhance the efficiency of contextual advertising with an ignorable decrease on effectiveness.

2.2. Keywords Recommendation

Generally speaking, keywords recommendation refers to finding the relevant keywords to a given target for some application. According to the type of the target, two major keywords recommendation problems are Web page keywords recommendation (also called keywords extraction) and related keywords recommendation.

2.2.1. Web Page Keywords Recommendation. As an important part of contextual advertising, Web page keywords recommendation is indispensable. In this process, valuable advertising keywords are automatically found to match the ads. It is obvious that the more accurate and valuable these found keywords are, the more relevant the ads will be delivered on the Web pages. Two kinds of approaches for Web page keywords recommendation are widely used currently. One is supervised learning and the other is unsupervised learning.

In supervised learning, a set of example pages that have been labeled with keywords by human editors are given as the training data. Features of each word should be carefully selected. There are several approaches, such as the traditional TF $\times$ IDF model, GenEx system [Turney 2000], the KEA system [Fang et al. 2005], and Yih's approach [Yih et al. 2006]. GenEx system is one of the best known programs for keywords recommendation. It is a rule based approach with 12 parameters tuned and is well used for pure textual content such as journal articles, email messages. However, keywords recommendation for Web pages should be more considered. Web pages contain various content structure with all kinds of multimedia information. Features of Web page should be taken into consideration to train the supervised learning model. The improved KEA [Turney 2003] and Yih's approach [Yih et al. 2006] bring various Web related features such as the meta data, URL, anchor, and so on. In the experiment, we select Yih's approach as the baseline keywords recommendation approach and more details will be presented in section 4.3.1. In the work of [Ravi et al. 2010], candidate bid phrases are generated through a translation model and then well-formed ones get ranked up by a language model.

The general idea of unsupervised learning is to create some appropriate formulas based on the features similar with the supervised learning to score these candidate keywords. Those who have scores in top k are recommended as keywords. In the work of [Litvak and Last 2008], the authors proposed to use HITS algorithm [Kleinberg 1999] to get the importance of the blocks (words, phrases, sentences, etc.) in a lexical or semantic graphs extracted from text documents. On the other hand, Matsuo proposes a new approach of unsupervised learning based on the co-occurrence information to optimize the recommendation result [Matsuo 2003]. Moreover, some approaches trying to enrich the target Web page content to improve the performance are also proposed, such as the work of [Ribeiro-Neto et al. 2005].

Although these approaches are effective in the experiments and have been widely used, a problem remains that these approaches highly depend on rich structure or content of the target Web pages. Thus for the short-text Web pages, these approaches can hardly provide high performance.

2.2.2. Related Keywords Recommendation. Given a target keyword, related keywords recommendation refers to finding the relevant keywords, such as synonyms, semantically relevant phrases, and some rewrite from the target keyword. It has been widely used in the field of information retrieval and search advertising.

In the information retrieval field, query expansion or query substitution is an important topic. The raw user queries will be processed to be substituted by one or a list

of keywords to obtain better search results [Jones et al. 2006; Mitra et al. 1998]. Much has been done about query substitution. Boldi et al. [Boldi et al. 2009; Boldi et al. 2008] make use of the user search session data to build a query flow graph and then use the random walk on the graph to get the related queries. Besides the session data, the search engine click through data is used for mining the similarity between queries in the work of [Cao et al. 2008].

There are also many works about related keywords recommendation for a given keyword in the search advertising. In search advertising, broad match helps indirectly match the user queries with the advertising bid keywords. Most of state-of-the-art matching algorithms expand the user query using a variety of external resources, such as Web search results [Abhishek and Hosanagar 2007; Joshi and Motwani 2006; Radlinski et al. 2008], page and ad click through data [Antonellis et al. 2008], search sessions, taxonomy [Broder et al. 2009] or concept hierarchy [Chen et al. 2008]. In addition, Radlinski et al. [Radlinski et al. 2008] consider more about the feasibility of matching ads and the search engine revenue. Wang et al. [Wang et al. 2009] improve the efficiency of query expansion for sponsored search by proposing a novel index structure and adapting a spreading activation mechanism.

2.3. Application of Wikipedia

Wikipedia is a free online encyclopedia in which large set of concepts are well expressed by experts and volunteers. It provides a considerable knowledge base, covering areas such as art, history, society, and science. Wikipedia is considered as an ideal knowledge base for not only the readers and researchers to look up knowledge but also the modern data mining systems to find auxiliary data to improve performance. Specifically, the articles of each Wikipedia entity contain a detailed explanation from various aspects. Moreover, the content of these articles are organized in well-structured format. This advantage can help automatic learning systems easier fetch information of entities. Furthermore, lots of links in the corpus of entities can imply a semantic relationship between linked entities, which can help automatic concept recognizers find related information.

Because of the diversity of content and the structured information [Medelyan et al. 2009], Wikipedia has attracted more and more researchers taking these advantages on the typical topics. Besides the application on keywords recommendation as introduced in this paper, many improvements has been achieved on other areas. P.Schönhofen [Schönhofen 2006] exploits the titles and categories of Wikipedia articles to identify the topics of documents. In the work of [Carmel et al. 2009], Wikipedia is applied to enhance cluster labeling and the authors claim that using Wikipedia entities to label each cluster outperforms using keywords directly in the text. Wang et al. improve text classification by enrich the document with the entities of Wikipedia [Wang and Domeniconi 2008; Wang et al. 2009]. Hu et al. map the target to Wikipedia thesaurus and use the entity content and links to enhance the query intent identification [Hu et al. 2009] and text clustering [Hu et al. 2008]. Yu et al. evaluate ontology based on categories in Wikipedia [Yu et al. 2007].

3. PRELIMINARIES

Before we discuss our algorithm, we first introduce some basic materials to set the stage for further discussion.

3.1. Topic-Sensitive PageRank

The PageRank algorithm [Brin and Page 1997; Page 1997; Brin et al. 1998] is a widely used algorithm to obtain the *static* quality of Web pages based on the link graph. The basic idea is to perform a random walk on the link graph and propagate the quality

score of a Web page to the ones it is linked to. Formally, the PageRank can be characterized by the following iteration,

$$\vec{R}_{m+1} = (1-d)\vec{B} + dG \cdot \vec{R}_m, \tag{1}$$

where $\vec{R}_m = [r_1^{(m)}, \cdots, r_n^{(m)}]^T$ is a vector of PageRank of the whole indexed Web pages on the mth iteration and $r_i^{(m)}$ stands for the PageRank score of the Web page i. Moreover, $\vec{B} = [\frac{1}{n}]_{n\times 1}$ is the damping vector and the decay factor α limits the effect of the propagation of PageRank. The matrix G represents the row-normalized adjacency matrix of the link graph, i.e., if there is an edge from vertex i to vertex j, then $G_{j,i} = \frac{1}{o_i}$, where o_i represents the out-degree of vertex i.

The topic-sensitive PageRank proposed in [Haveliwala 2002] extends PageRank by allowing the iteration process be biased to a specific topic. Specifically, the damping vector $\vec{B}$ is biased to the specific topic so that the Web pages of this particular topic are more likely to have high scores. The iterative scheme is the same as Formula 1 except the damping vector $\vec{B} = [b_1, \cdots, b_n]^T$ with the damping value b_i indicating the relevance of page i to the topic in question. It is easy to see that the propagation process is biased by the damping vector $\vec{B}$ at each iteration. Consequently, pages with higher damping values can propagate higher scores to their neighbors in the link graph.

3.2. Wikipedia Graph

In this section, we consider the construction of the Wikipedia graph. Firstly, we take each entity as a vertex in the graph. Then we aim at connecting two entities with an edge of the graph if they are semantically related. To this end, we notice that there are many hyperlinks in each Wikipedia article linking to the pages of other articles of Wikipedia entities. These hyperlinks are just potential edges for the graph we want to construct, because many Wikipedia articles link to (other articles about) dates and regions or other entities that are general but otherwise semantically unrelated. To address this issue, we make use of the entity category information. Specifically, we remove the edges between the entities of different first-level categories in the Wikipedia category tree. This way, there are only edges linking the entities in the same topic. Therefore, the Wikipedia graph reduces to κ subgraphs, where κ stands for the number of first-level categories, specifically $\kappa = 10$ for a Wikipedia version in Jan. 2010. Additionally, we also consider weighting the edges by the number of links between two entities. This is reasonable because more edges from entity i to entity j means entity j is more related than other neighbors of entity i. Therefore, given the whole set of Wikipedia articles, we can construct a directed graph, which is called *Wikipedia Graph*. In the rest of the paper, we will denote the ro-w-normalized Wikipedia graph link matrix as $\mathcal{G}$, which is a $n \times n$ matrix with n stands for the size of the entity set. Element $\mathcal{G}_{i,j}$ of the matrix is given by

$$\mathcal{G}_{i,j} = \frac{\eta(j,i)}{\sum_{k=1}^{n} \eta(j,k)} \tag{2}$$

where $\eta(j,i)$ denotes the edge number from entity j to entity i. Figure 3 gives an illustration of a Wikipedia subgraph.

4. CONTENT & ADVERTISEMENT-SENSITIVE PAGERANK

4.1. Problem Definition

The problem we need to address is: Given a target Web page p as the input, particularly, one with short-text content. The output should be k ranked keywords for p, required to be both relevant to the content of p and valuable for advertising.

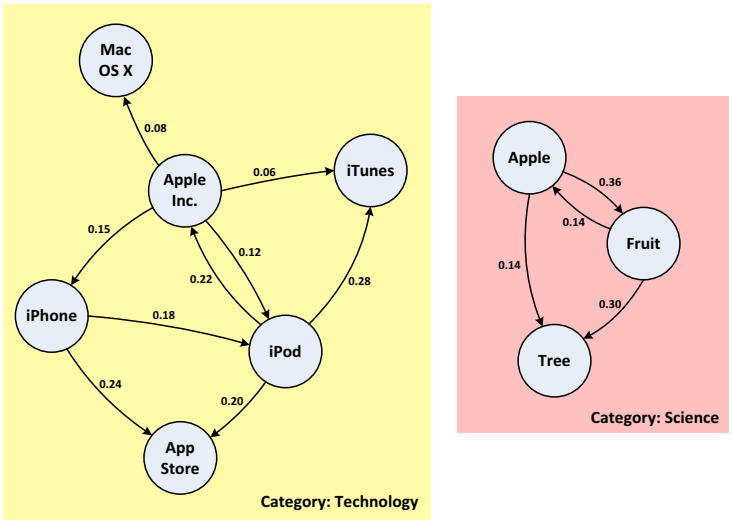

Fig. 3. An illustration of a Wikipedia subgraph.

4.2. Algorithmic Details

Here we describe the details of our algorithm of content & advertisement-sensitive PageRank on the Wikipedia graph.

As our problem focuses on short-text Web page, the textual information of these pages themselves tends to be very limited. This information, however, is still an indispensable part for the proposed algorithm. Specifically, the textual content of p is first parsed and the Wikipedia entities occurring in p are identified with the same approach in [Wang and Domeniconi 2008]. Then those entities are scored using traditional approaches such as, in our experiment, the supervised learning approach similar to Yih's work [Yih et al. 2006]. The score, defined as *content relevant score*, indicates the relevance of the entity to the content of p and measures the possibility of the entity to be a keyword of p.

Secondly, the output keywords of our algorithm should be valuable for advertising. Intuitively, the frequency of an entity name in some advertisement content can be a criterion to measure its advertising value. Therefore, another score on each entity is given, which we term as *advertisement relevant score*, defined as the percentage of occurrences of each entity name in a given set of ads. This score measures how likely the entity will occur in advertisement texts and can be used to retrieve the corresponding advertisement. In addition, this score is not based on the input target Web page, and thus can be calculated offline. As a consequence, every entity has two topic scores: the content relevant score and the advertisement relevant score.

Thirdly, we use the two scores in the content & advertisement-sensitive PageRank iterations. In this process, the entities that are related to the content of target Web page and valuable for advertising will get higher score because of the two scores on content and advertisement bias. The neighbors of these entities will also get a higher score because of the propagation process of PageRank score.

Finally, we rank and choose k entities with the highest content & advertisement-sensitive PageRank score as the output of our algorithm. The overall steps of the algorithm is depicted in Figure 4.

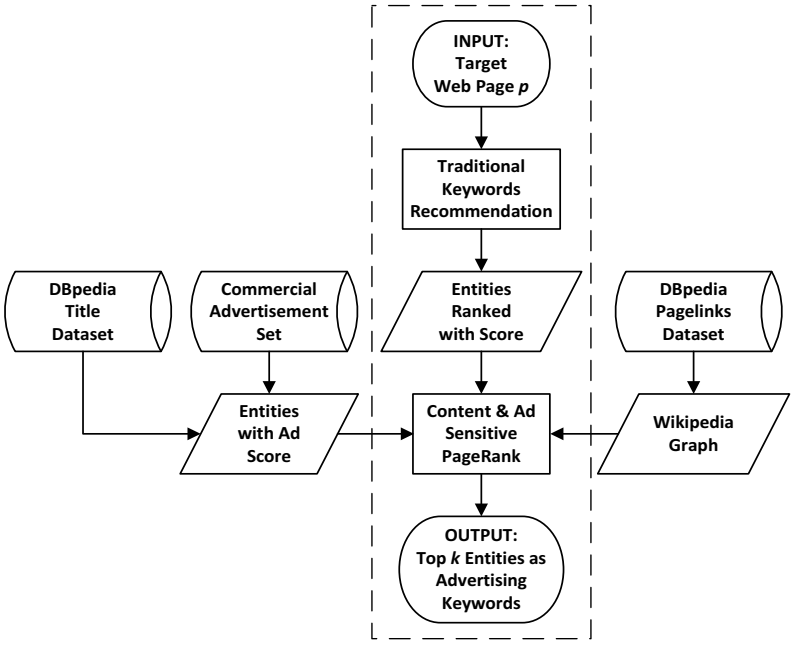

Fig. 4. Algorithm flow chart. The online process is in the dotted box and the offline one is outside.

We next describe the details of the content relevant score and the advertisement relevant score in section 4.3. Utilizing the two scores as the damping vectors, the work of content & advertisement PageRank is described in section 4.5.

4.3. Damping Vector Setup

Considering that the recommended keywords should be both relevant to the target Web pages and valuable for advertising, we define two factors for each entity, corresponding to two types of damping vectors in the expression of topic-sensitive PageRank.

4.3.1. Content Damping Vector Setup. Given the target Web page p, for each entity i, its content damping value c_i stands for the relevance between entity i and the content of p.

We used a supervised learning approach to score each entity and determine whether an entity has a score high enough to be treated as a keyword. In our approach, we implement the approach of [Yih et al. 2006] to generate the feature vector for each entity in p. Firstly, p is parsed and Wikipedia entities in p are extracted. Then these entities in p are scored by a regression approach. The regression approach is implemented by *Support Vector Machine*, which has been trained with a large set of Web pages with keywords extracted by human editors. The entity with a higher score implies that it is more important to p. The features we selected for regression are listed in Table I.

The last feature in the list, QueryLog, which is proposed in Yih's work [Yih et al. 2006], appears to be a novel feature to help improve the performance of supervised learning. It is claimed that the entities that occur in the user query are more likely

Table I. Features selected for supervised learning

Name	Type	Description
URL	Boolean	Whether it is in the page URL
Title	Boolean	Whether it is exactly the title
Headline	Boolean	Whether it is exactly the headline
Anchor	Boolean	Whether it is a the anchor text
TF	Double	The frequency of the entity in the page (normalized)
Link	Boolean	Whether it is part of a hyperlink of the page
PartLink	Boolean	Whether part of the entity is part of a hyperlink of the page
Meta	Boolean	Whether it is in the meta text
Span	Boolean	Whether it is in the span text
OneCapt	Boolean	Whether the first word of the entity is capitalized
AllCapt	Boolean	Whether each word of the entity is capitalized
Length	Double	The string length of the entity (normalized)
QueryLog	Boolean	Whether the entity is in the query log from AOL

to be keywords since the users use them in search engine to retrieve some pages they want. In our experiment, we use the query log from AOL[8].

These features are of different importance, for example, an entity appears in the title is more likely to be a keyword than an entity just appearing somewhere in the content body. Therefore, we weight those features and tune the weight to obtain the best performance.

4.3.2. Advertising Damping Vector Setup. Using advertisement-sensitive PageRank, we can obtain more commercial entities which are suitable for the advertising keywords. The entities which are more relevant to advertisement topic should have higher advertisement damping value. In our approach, we record the frequency of each Wikipedia entity i in the text of an advertisement set, defined as a_i. For the calculation of the PageRank iteration, we let a_i be the damping value biased to advertisement for each entity i.

4.4. Two Topic-Sensitive PageRanks

Based on the topic-sensitive PageRank and the two topic relevant scores, we can construct two topic-sensitive PageRanks: *content-sensitive PageRank* and *advertisement-sensitive PageRank*. Given the content relevant score c_i for each entity i to the target Web page p, we can obtain the *content damping vector* $\vec{C}$. The iteration formula of content-sensitive PageRank is

$$\vec{R}_{m+1} = \alpha\vec{C} + (1-\alpha)\mathcal{G} \cdot \vec{R}_m, \tag{3}$$

where $\vec{R}_m = [r_1^{(m)}, \cdots, r_n^{(m)}]^T$ is the vector of the PageRank scores of all the entities on mth iteration, the parameter α controls the impact of the content relevant score to the content-sensitive PageRank value[9]. Similar to topic-sensitive PageRank on Web pages, this process can propagate the relevance scores from the seed entities with high content relevant scores to other relevant entities even if they do not occur in the target Web page. As a result, we can enrich the keywords set of the target Web page with the help of content-sensitive PageRank.

Similarly, given the advertisement relevant score a_i for each entity i, we can obtain the *advertisement damping vector* $\vec{A}$. The iteration formula of advertisement-sensitive

[8]AOL search data mirrors. http://www.gregsadetsky.com/aol-data/

[9]Here α corresponds to $1-d$ in Formula 1. Using this notation makes it seamless to merge the two topic-sensitive PageRanks into a multi-topic-sensitive one, as will be discussed latter.

PageRank is

$$\vec{R}_{m+1} = \beta\vec{A} + (1-\beta)\mathcal{G} \cdot \vec{R}_m, \tag{4}$$

where β is the parameter that controls the impact of the advertisement relevant score to the advertisement-sensitive PageRank value. Likewise, the entities that are related to the advertisement topic are more likely to get higher scores after every iteration by using the impact of advertisement damping vector $\vec{A}$. In addition, neighboring entities can share those scores, which may also be relevant to the advertisement. Thus, the advertisement bias is incorporated into the random walk process.

4.5. Content & Advertisement-Sensitive PageRank Algorithm

As we emphasized before, the recommended keywords should be both relevant to the target Web page and valuable for advertising. These two requirements can be satisfied by the two topic-sensitive PageRanks introduced above. We propose an approach that combine the two different topic-sensitive PageRank to simultaneously address the above two requirements. The iteration formula combining the two topic-sensitive PageRanks is

$$\vec{R}_{m+1} = \alpha\vec{C} + \beta\vec{A} + (1-\alpha-\beta)\mathcal{G} \cdot \vec{R}_m \tag{5}$$

In the above equation, there are two damping vectors: $\vec{C}$, biased to the target Web page content, and $\vec{A}$, biased to the advertisement topic. Intuitively, in each iteration, the PageRank of entity both relevant to the target Web page content and the advertisement topic will get a relatively high score boost by the factor $\alpha c_i + \beta a_i$, and each entity will also distribute its score to its neighbors. Therefore, entities that are relevant to both the content of target Web page and the advertisement topic can obtain higher scores after the convergence of iteration process.

4.6. Computational Details

4.6.1. Initial PageRank Vector Setup. As is well-known the convergence of PageRank will not depend on the initial value for each entity (the element of the start vector $\vec{R}_0$). The number of iterations and thus execution time, however, are greatly affected by the initial values. Thus an appropriately-chosen initial value for each entity will improve the efficiency of the PageRank iteration process.

We propose to set the initial vector as

$$\vec{R}_0 = \alpha\vec{C} + \beta\vec{A} \tag{6}$$

where α and β are the same parameters in Formula 5 while $\vec{C}$ and $\vec{A}$ have been determined in section 4.3.1 and section 4.3.2. In our experiment, it takes about 25% less time to get convergence, compared with setting initial vector with uniform value.

4.6.2. Computational Complexity. In this section, we discuss the computational complexity of content & advertisement-sensitive PageRank. The computation process $(1-\alpha-\beta)\mathcal{G} \cdot \vec{R}_m$ theoretically takes an $O(n^2)$ time. Since we have removed the cross-category edges when constructing the Wikipedia graph, we need not traverse all the entities in each iteration process, instead we just traverse the subgraphs in the categories of the seed entities. Therefore, in each iteration, $O(\overline{n})$ vertices are involved, where $\overline{n} = n/\kappa$ stands for the average number of entities in each subgraph, κ has been mentioned in section 3.2. Furthermore, Wikipedia graph is a sparse graph with each entity having on average 18.3 in-edges, thus $\mathcal{G}$ is a sparse matrix where each row has on average no more than 18.3 non-zero number. Taking advantage of this, we only record the in-edge neighbors for each entity and thus the calculation process reduces to an $O(\overline{n} + \overline{n} + 18.3\overline{n})$ time.

Given the initial PageRank vector, the number of iterations is still dependent to the two parameters α and β. Generally, the larger $(1-\alpha-\beta)$ is, the more slowly the convergence is. In our experiments, six iterations to reduce the fluctuation of PageRank value $\triangle r_i^{(m)} = |r_i^{(m+1)} - r_i^{(m)}|$ under 10^{-4} when setting $\alpha = 0.85$ and $\beta = 1.5 \times 10^{-5}$. In our experiment, the average real run-time for each test case is 8.26 seconds (Java Platform, Windows, 2GB memory, 2.6GHz). Furthermore, the efficiency can be improved with the optimization work discussed in the next subsection.

4.6.3. Optimization for Efficiency Improvement. Several optimizations can be incorporated for accelerating the computation. In this subsection, we discuss the optimization work which has been implemented in our experiment.

In PageRank calculation, it is time consuming for updating PageRank value of the whole entity set. For our model, the number of entity with non-zero initial value is extremely low (20, in our experiment). Thus in each calculation, those entities with zero value and without non-zero neighbor can be ignored. We just consider involved entity set, in which the entity PageRank value will be changed. Before the ith iteration of the computation, define the set of entities with non-zero PageRank score as S_{i-1}. It is clear that in this step of computation, only the entities in S_{i-1} or the ones with an in-edge from S_{i-1} will have a non-zero entry. Therefore, in the ith iteration of the computation, we only consider S_i in the computation instead of all the entities in the WikiGraph. In our experiment, the average real run-time for each test case is reduced to 2.95 seconds with this optimization.

5. EXPERIMENTS

This section mainly discusses the experimental results for our proposed algorithm, including the data preparation, comparisons with existing algorithms, evaluation metrics, performance results, and further discussions.

5.1. Data Preparation

5.1.1. Wikipedia Graph. DBpedia[10], according to its description, is a community effort to extract structured information from Wikipedia and to make this information available to the general public. The structured information can be downloaded from its Web site.

In our experiment, we take *Pagelinks* dataset with the date of Sept. 2009. It has 81.83 million triples and each triple represents a relation that the first entity has a page link to the second entity, resulting an initial graph with 81.83 million directed edges and 9.54 million entities. We refine the graph by removing the entities without out-edges and the ones which have some characters with ASCII $>$ 127. As mentioned in section 3.2, we also removed the cross-category edges with the help of *Article Categories* and *Categories(Skos)* data set on DBpedia. With this preprocessing, the entity number reduces to 3.12 million and the edge number 57.29 million, as is shown in Table II.

Table II. Data of Wikipedia Graph

Release Date	Vertice(entities)	Edges(pagelinks)	Average Out-degree
Sept. 2009	3.12 million	57.29 million	18.3

[10]DBpedia. http://dbpedia.org/

5.1.2. Advertisement Set. We use a set of 9 million textual ads, which are crawled by commercial search engine using the AOL query log data. Each advertisement consists of the title and the description of the ads, which contain up to 10 and 30 words, respectively. Both of the two sections of the ads are composed by the advertisers.

5.1.3. Training Data. We use 6422 human labeled Web pages as our training pages. We extract the Wikipedia entities from the training pages and construct the vector for each entity with the feature described in Table I. Since those entities have been labeled, we can use these vectors with labels to train a classifier.

5.1.4. Test Data. We use the corpus of ODP Web pages as test pages in our experiments. The number of sampled test pages should be determined so that the experiments can convincingly demonstrate the difference of performance among algorithms compared. In addition, the test pages should cover more topics so as to be representative. In our experiments, we generated two test data sets. The first data set consists of 100 randomly selected short-text[11] ODP Web pages, called *short-text Web page set*. The second one consists 103 Web pages, with 50 ones in short-text and the other 53 in moderate or long text. We call the second test data set *overall Web page set*. These Web pages cover the topics of business, agriculture, art, computers, entertainment, automobile, sports, Internet, life product, medicine, music stars, and so on.

5.2. Algorithms Used for Comparison

To demonstrate the effectiveness of our approach, we use some other approaches as the baseline or control in our experiments. All the approaches are listed below, including our proposed approaches.

1. **TF Counting (TF).** We simply take out the k entities with the highest frequency in the target Web page p.
2. **Supervised Learning (SL).** By training a SVM with the training pages labeled by human editors, we can obtain the relevant score for every entity in the target Web page p from SVM. Then we choose k entities with the largest score to be the result keywords [Yih et al. 2006]. This approach can be considered as a pre-processing step the of content & advertisement-sensitive PageRank. The scores for the entities will be used as the content damping vectors of the iteration process of the content & advertisement- sensitive PageRank.
3. **Co-occurrence in Ads (CA).** We use a process to mine the co-occurrence of two entities in one advertisement content. Generally, if entity A and B occur in the same advertisement for more than t times, then we claim that A and B are friends. Given the target Web page p, we find the related entities to p with the most friends in p as the result.
4. **Query Click-through Bipartite Gragh (QCBG).** Here we implement one traditional query expansion approach for the related Web page keywords recommendation. This approach is based on query clustering using both query content and the query-URL bipartite graph [Cao et al. 2008; Wen et al. 2001]. We use the AOL query click-through data to build the query-URL bipartite graph. After query clustering, we can generate expansion for each target query. For the target Web page p, a candidate keyword set is obtained by the expansion on the seed keywords of p. Then, the closest k keywords to p are finally recommended, where the keywords are represented by vectors used in clustering and p is represented by merging the seed keywords vectors.

[11] with less than 100 terms.

5. **Content-sensitive PageRank (CPR).** We have proposed this approach in section 4.4. As this algorithm does not use any advertisement information, it is used to compare the commercial impact with Algorithm 6.
6. **Content & Advertisement-sensitive PageRank (CAPR).** We have proposed this approach in section 4.5. It is our key approach in this paper.

5.3. Evaluation Measures

The input of the experiment is a target Web page p and the output is k keywords to p. For the gold-standard of the evaluation work, we invited five colleagues to judge the relevance of each page-keyword pair as follows.

— *Relevant and advertisable.* The keyword is relevant to the content of target page and also it has a possibility to be valuable for advertising, scored as 1.
— *Otherwise.* The keyword is not considered as relevant to the content of target page or it is impossible to be used as advertising keyword, scored as 0.

Each page-keyword pair has at least two human judges. After the judgment work, we average the scores for each page-keyword pair. The scores can be interpreted as the possibility of relevance and advertisability[12]. Then we evaluate the performance of the algorithms using the evaluation measure below.

In our experimental studies, we use P@n as evaluation measure. Precision at position n (P@n) is defined to be the fraction of the top-n retrieved keywords that are relevant [Baeza-Yated et al. 2008].

$$P@n = \frac{\sum_{i=1}^{n} \tau_i}{n} \tag{7}$$

In Formula 7, τ_i denotes the average rate score for the pair of the target Web page and the ith recommended keyword. Since we not only accept the good keywords occurring in p, but also the related ones, there is no good measure to evaluate the recall of each approach.

5.4. Experimental Results

Our experiments are divided into five parts. In the first part, we normally do the keywords recommendation with four approaches TF, SL, CPR, and CAPR. The output keywords can be both in-page keywords and leveraged keywords, called *universal keywords*. In the second part, we focus on the leveraged keywords recommendation (without in-page keywords in the result), using approaches implemented from algorithm CA, QCBG, CPR, and CAPR. The third part reports the impact of the parameters of CAPR. In the fourth part, we present a case study to analyze the ability of CAPR for dealing with the ambiguity of Wikipedia entities. In the last part, we demonstrate the recommendation performance against the word number of the target Web pages.

5.4.1. Universal Keywords Recommendation. Universal keywords recommendation judges the practical effectiveness of each algorithm, both in-page and leveraged keywords are recommended. For every target Web page p, each approach provides a result of 20 keywords for p. We demonstrate the overall precision of the four approaches in top 5, top 10, top 15, and top 20 keywords recommended. Here we compare the results of the four approaches: TF, SL, CPR, and CAPR.

First we use the short-text Web page set for testing. The performance of the four approaches is shown in Table III and Figure 5.

[12]Term 'advertisability' has been used in the work of [Pandey et al. 2010].

Table III. Precision in top k of the results of the four approaches on short-text Web page set.

	TF	SL	CPR	CAPR
Top5	0.4260	0.4195	0.5708	0.5751
Top10	0.3363	0.3248	0.5428	0.5580
Top15	0.2661	0.2653	0.4488	0.4753
Top20	0.2128	0.2203	0.4019	0.4514

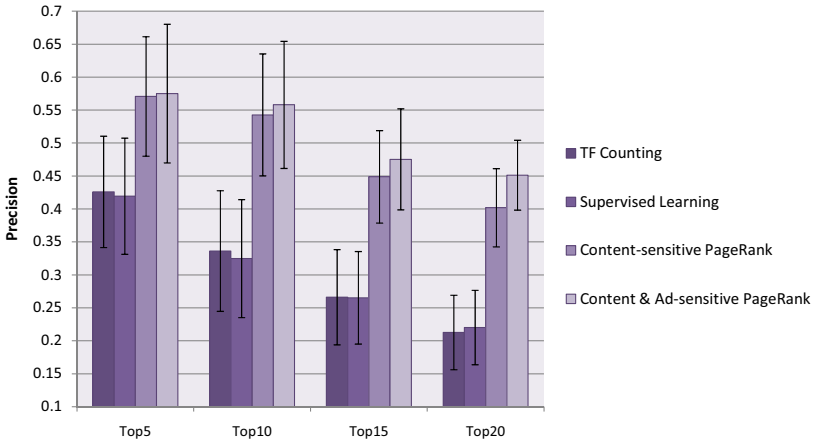

Fig. 5. Precision in top k of the results of the four approaches on short-text Web page set.

Several observations are interesting to note: (i) CAPR shows significant improvement over other approaches on the short-text Web page set. Compared with SL, CAPR has an improvement of 37.09%, 71.82%, 79.15%, and 104.94% on top 5, 10, 15, and 20 respectively. It verifies that combining content & advertisement-sensitive PageRank helps to find leveraged keywords which is more relevant to the target Web page than in-page keywords. (ii) Somewhat unexpectedly, the traditional supervised learning approach SL is no more effective than the TF counting approach, which is very simple and considered to be the baseline in the experiments. The possible reason is that SL over-emphasizes the page content and structure, which is sparse and much diverse in short-text Web pages. (iii) For most short-text target Web pages, the traditional approaches (TF, SL) could not even provide more than 15 keywords and the precision of the keywords is not so good, less than 45%. This reveals the problem of traditional keywords recommendation on short-text Web pages. In section 5.4.5, we will give a panoramic view of the performance of these approaches against the size of target Web page. (iv) Compared with CPR, CAPR has an improvement of 0.76%, 2.81%, 5.90%, and 12.32% on top 5, 10, 15, and 20 respectively, which verifies the effectiveness of incorporating advertisement bias in CAPR.

Besides the short Web pages, we also demonstrate the performance of these approaches on overall Web pages set, not just short-text Web pages. The performance is shown in Table IV and Figure 6.

As is shown in Figure 6, CAPR performs the best in all the comparison and has an improvement of 5.30%, 14.29%, 20.18%, and 29.92% on top 5, 10, 15, and 20 over SL respectively. It proves that CAPR also works well on the long-text Web pages even though the improvement is not so significant as on short-text Web pages and advertisement bias also helps improve the performance. Moreover, SL performs better than the baseline TF, which indicates that SL works well mainly on long-text Web pages.

Table IV. Precision in top k of the results of the four approaches on overall Web pages set.

	TF	SL	CPR	CAPR
Top5	0.5782	0.6356	0.6622	0.6693
Top10	0.4792	0.4921	0.5443	0.5624
Top15	0.4238	0.4284	0.4897	0.5149
Top20	0.3693	0.3822	0.4622	0.4965

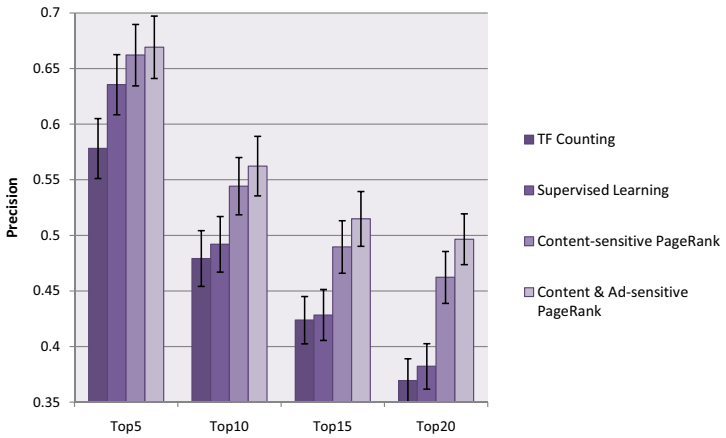

Fig. 6. Precision in top k of the results of the four approaches on overall Web pages set.

Compared with those on short-text Web pages, the performance of each algorithm has lower standard error, shown as error bars.

Since CAPR recommends both in-page keywords and leveraged keywords, we also did an analysis of the in-page keywords proportion in the recommendation of CAPR, as is shown in Table V and Figure 7. From the result we have following observations: (i) As the number of recommended keyword increases, in-page keyword proportion decreases. For each target Web page, the in-page keyword number has an upper limit. Thus after the most suitable in-page keywords are recommended, more and more leveraged keywords are more likely to be recommended than the remaining in-page keywords. (ii) On overall Web pages set, the in-page proportion is smaller than that on overall Web pages set. This is also reasonable because there are fewer in-page keywords in the short-text Web pages and leveraged keywords are more likely to occur in the result.

Table V. Proportion of in-page keywords in the recommendation of content & advertisement-sensitive PageRank on overall Web pages set and short-text Web page set.

	Overall Web Page Set	Short Web Page Set
Top5	0.9089	0.6752
Top10	0.7366	0.5373
Top15	0.6640	0.4374
Top20	0.6279	0.4125

As a case study of the comparison for the approaches of TF, SL, and CAPR, we here provide the performance of them on a specific short-text test Web page case of *ION Me-*

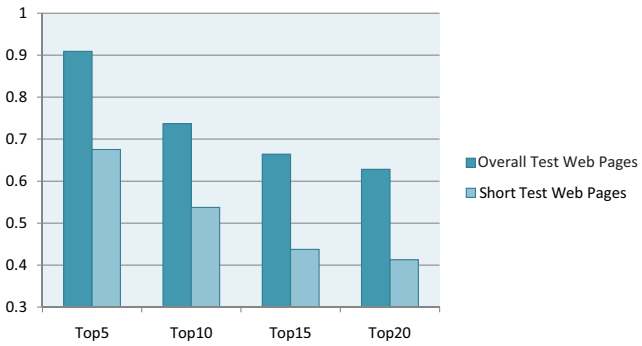

Fig. 7. Proportion of in-page keywords in the recommendation of content & advertisement-sensitive PageRank on overall Web pages set and short-text Web page set.

dia Networks[13]. The performance is shown in Table VI. From the performance result, we can know that TF hits 6 keywords, SL hits 6 keywords, and CAPR hits up to 11 keywords in the top 20, which dominates in the case study.

Table VI. 20 Keywords Recommended to Home Page of *ION Media Networks*.

	TF	SL	CAPR
1	**ION Media Networks**	**ION Media Networks**	**ION Media Networks**
2	**Television**	**ION Life**	**ION Life**
3	**ION Life**	**Television**	**Television**
4	Company	power	power
5	Life	Completed	Completed
6	Completed	Company	**Broadcasting**
7	power	Life	Company
8	**Sony Pictures Television**	CBS	**Comcast**
9	Twentieth Television	DTV	Qubo
10	RHI Entertainment	Bros	**NBC**
11	Entertainment	Video	Rock music
12	**NBC Universal**	Pictures	Buffalo, New York
13	Open Mobile	Coalition	**American Broadcasting**
14	Coalition	Universal	Touched by an Angel
14	Universal	Entertainment	**Fox Sports Net**
16	Pictures	**Sony Pictures Television**	**Talk radio**
17	Video	**Twentieth Television**	**Ion Television**
18	Bros	RHI Entertainment	**Cornerstone Television**
19	CBS	**NBC Universal**	Rochester, New York
20	DTV	Open Mobile	Wilmington, Delaware
Hit	6	6	11

5.4.2. Leveraged Keywords Recommendation. Here we investigate the performance only on the recommended leveraged keywords. Among the compared algorithms, CA, QCBG, CPR, and CAPR have the ability to recommend leveraged keywords of the target Web pages, simply by filtering out the in-page keywords in the results. Similar to the section 5.4.1, we first demonstrate the precision of the four approaches on top 5, 10, 15, and 20 keywords recommendation on short-text Web page set. The performance of those approaches is shown in Table VII and Figure 8.

[13]ION Media Networks. http://www.ionmedia.tv/

Table VII. Precision in top k of the results of the four approaches on short-text Web page set.

	CA	QCBG	CPR	CAPR
Top5	0.3025	0.4320	0.5027	0.5033
Top10	0.1992	0.3430	0.4367	0.4534
Top15	0.1470	0.2967	0.4210	0.4403
Top20	0.1219	0.2680	0.3696	0.4119

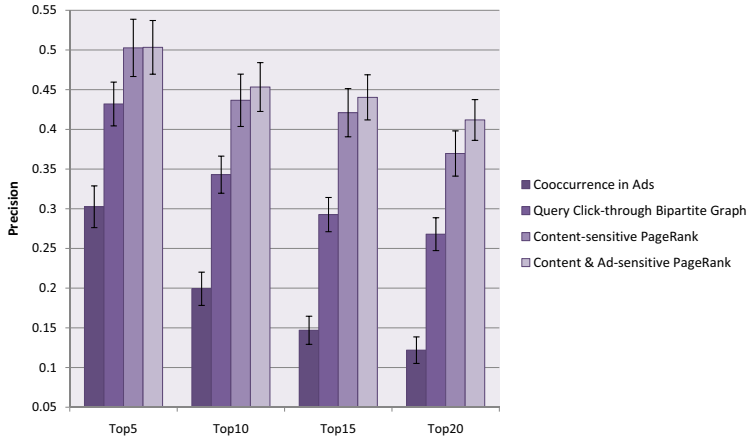

Fig. 8. Precision in top k of the results of the four approaches on short-text Web page set.

From the results, we can know that (i) CAPR gives a high performance in the work of leveraged keywords recommendation. It has an improvement of 16.52%, 32.18%, 50.46%, and 53.69% on top 5, 10, 15, and 20 against QCBG. Its performance of leveraged keywords precision could even compared to the universal keywords recommendation performance. (ii) On the contrary, CA is not as suitable for related advertising keywords recommendation. For one reason, the short-text Web page makes it much difficult to determine the topic or the keywords from which to get the leveraged keywords; for another reason, there is so much noise in the advertisement text, such as the misleading and ambiguous expression in advertisement text and the unbalance of the entities frequency in the advertisement set, which misguide the relation of two entities in the mining process of co-occurrence in ads. (iii) QCBG performs a little worse than CAPR on top 5 results. However, as the recommended keyword number increases, the performance gap between these two algorithms increases significantly. This is because the user queries are highly diverse and thus query expansion will possibly import some irrelevant keywords. (iv) Furthermore, the result also demonstrates that advertisement bias helps improve the performance of leveraged keywords by 0.53%, 3.82%, 5.59%, and 11.43% on top 5, 10, 15, and 20. From the performance figures, we can conclude that the ability to find the leveraged keywords is much important in the field of advertising keywords recommendation especially on the short-text Web page set.

We also make a universal keywords recommendation for comparison on overall Web page set, not just short-text Web pages. The performance is shown in Table VIII and Figure 9.

As is shown in Table VIII, CAPR dominates in each comparison and has an improvement of 36.67%, 71.10%, 77.69%, and 71.34% on top 5, 10, 15, and 20 over QCBG respectively. From the result we can know CAPR still works well on the long Web pages

Table VIII. Precision in top k of the results of the four approaches on overall Web page set.

	CA	QCBG	CPR	CAPR
Top5	0.2080	0.3980	0.5375	0.5440
Top10	0.1813	0.3178	0.4938	0.5438
Top15	0.1686	0.2891	0.4392	0.5137
Top20	0.1611	0.2772	0.4167	0.4750

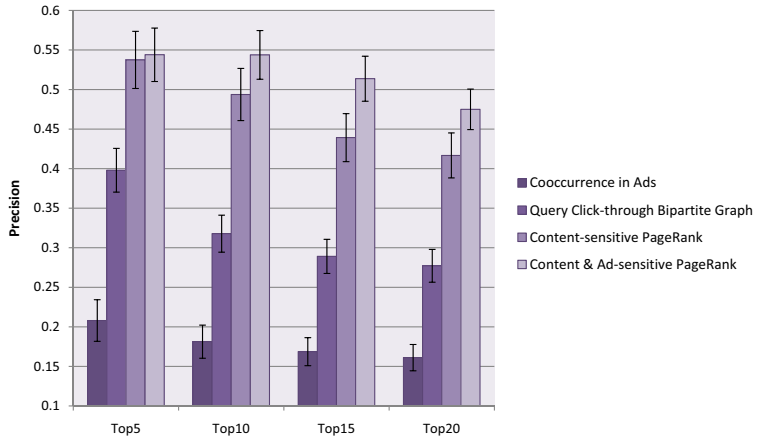

Fig. 9. Precision in top k of the results of the four approaches on overall Web page set.

and advertisement bias also helps improve the performance by 1.21%, 10.14%, 16.96%, and 13.99% on top 5, 10, 15, and 20 respectively. Thus it verifies that CAPR is the most competent to recommend leveraged keywords to the Web pages.

5.4.3. Parameter Tuning. As is shown in Formula 5, there are two parameters, α and β, in CAPR that need to be tuned. In this section, we focus on the impact of the parameters on the performance of the algorithm.

In our experiment, the two parameters are tuned separately. First, we fix $beta$ to 5.0×10^{-5} and tune α from 0.30 to 0.98 (some preliminary experiment has shown that this area produces the best result). Secondly, we fix α to 0.9, which is sensitive to our Wikipedia graph, tune β from 1.0×10^{-5} to 1.0×10^{-3}. The precision on Top 20 recommended keywords can be depicted against α and β, which is shown in Figure 10.

From Figure 10, it can be noted that there is a tradeoff between the weight of content bias and the propagation of PageRank by tuning α and when β is larger than 2.0×10^{-5}, the performance reduces as β increases. Finally, we set $\alpha = 0.85$ and $\beta = 1.5 \times 10^{-5}$ to run CAPR. It is necessary to point out that α and β here have already been combined with the normalization factor of respect damping vectors, which explains the gap between the orders of magnitude α and β.

5.4.4. Noise Impact and Ambiguity Analysis. As has been discussed in section 4.2, the seed entities of propagation step are given by traditional recommendation approaches. Will the propagation step of CAPR amplify the errors made by those approaches if there are some irrelevant keywords, called *noise*, in the seed entity list? Moreover, in the application of Wikipedia, one common problem is the ambiguity of entity names. In the area of keywords recommendation, this problem still exists. For example, if a term

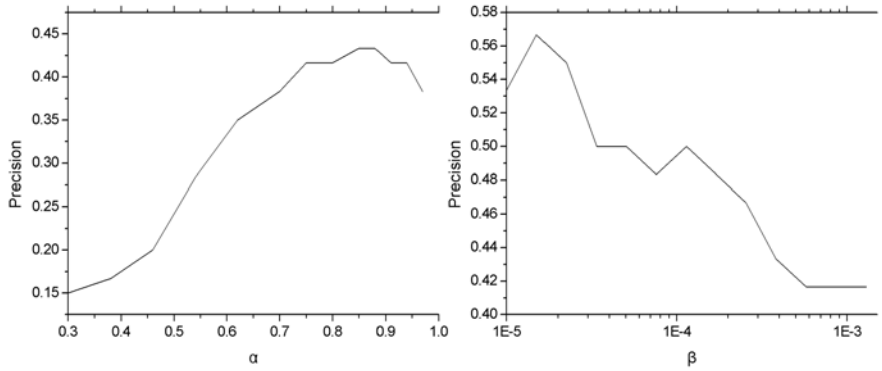

Fig. 10. Precision in top 20 of the results against the parameter α and β.

Apple occurs in the target Web page, whether it means the fruit or the Apple brand[14]. In Wikipedia, the term *Apple* will directly be matched to the one under fruit category. So if the term *Apple* in the page actually means the brand of Apple, the ambiguity problem exists.

However, from the experiments above, it is found that the noise and ambiguity problem do not seem to sinificantly reduce the performance of CAPR. We claim that topic-sensitive PageRank can reduce the impact of noise and ambiguity of Wikipedia entities. Here we provide a preliminary analysis. Given some seed entities, several of which may be noisey or ambiguous and can match different Wikipeida categories. Since we have removed the cross-category edges in the graph, the ambiguous entities cannot distribute much PageRank value to their neighbors unless the ambiguous entities outnumber the entities under the correct first-level category. For a concise example, for a target Web page about electronic product of Apple, given three seed entities: *Apple, IPod, and IPhone*, where *Apple*, as is mentioned above, is a noise in the seed entities. Here we present the top 20 recommended keywords from CAPR in Table IX.

Table IX. Top 20 keywords offered by CAPR from the seed entites: *Apple*, *IPod*, and *IPhone*, where Apple is a noise seed entity.

Seed entities: Apple, IPod, IPhone		
Result Top 20 Keywords:		
IPhone	**IPod**	Apple
E-mail	**Comparison of iPod managers**	**Apple Inc.**
Mac OS X	**ITunes**	**Steve Jobs**
IPod Touch	United States Dollar	Wired (magazine)
Macworld	**IPhone OS**	**Bluetooth**
ITunes Store	**Nokia**	**Flash memory**
Portable media player	**ARM architecture**	

From Table IX we can see that 17 keywords are in the topic of Electronic Products, 2 keywords are Wikipedia noise entities, only *Apple* belongs to the fruit topic. This example verifies the disambiguating ability of content & advertisement-sensitive PageRank.

[14]Apple Inc. http://www.apple.com/

5.4.5. Performance against Target Web Page Content Size. In the last part of our experiment, we analyze the performance against the content size of target Web pages. We make a comparison on TF, SL, and CAPR on the page size domain of 1 to 15104. Specifically, we divide the pages into 4 groups by their word numbers, the intervals of which are $1 \sim 99$, $100 \sim 399$, $400 \sim 799$, and 800+. The performance on the top 10 keywords recommended of three two algorithms on each group of pages is in Table X and Figure 11.

Table X. Precision in top 10 of the results of the three approaches against the page size.

Page Size	TF	SL	CAPR
0 to 99	0.3363	0.32.48	0.5580
100 to 399	0.5647	0.6294	0.6588
400 to 799	0.5778	0.6815	0.6889
800 +	0.6571	0.6786	0.7357

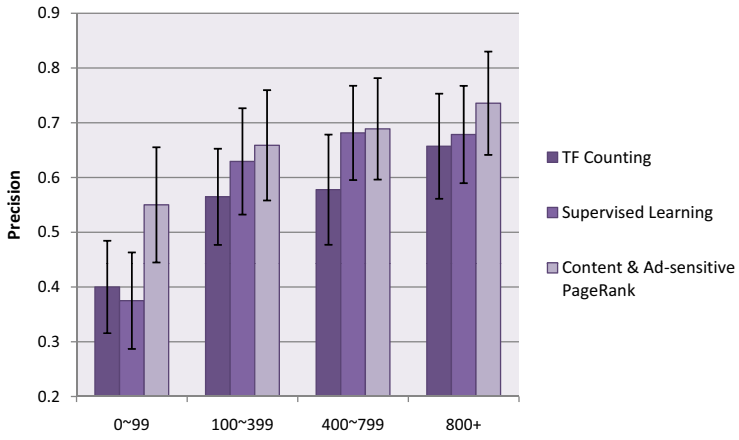

Fig. 11. Precision in top 10 of the results of the three approaches against the page size.

From the result we observe (i) On average, the performance of keywords recommendation improves as the content size of target Web page increases while when the page content size is especially small, for example less than 100, the traditional approaches (TF and SL)for keywords recommendation does not work well (less than 40% in our experiment). (ii) Our approach CAPR still works well on the short-text pages (55.01% in our experiment), making an improvement of 91.30% on SL. (iii) As the content size increases, traditional approaches show large variation (SL varies from 37.50% to 67.86%, increasing by 80.96%) but our algorithm indicates more robustness and less sensitivity to the content size of the target Web pages (from 55.01% to 73.57%, increasing by 33.74%).

5.5. Discussion

Based on the five parts of the experiment, we claim that Content & Advertisement-sensitive PageRank works well for the advertising keywords recommendation. It provides a significant improvement over several state-of-the-art approaches on short-text Web pages. For short-text target Web pages, the keywords recommended by traditional

approaches are always with noise. However, in the propagation step of our approach, the noise is reduced and the keywords in the main topics get higher ranks due to the propagation of Content & Advertisement-sensitive PageRank score. In addition, leveraged keywords which do not occur in but still relevant to the target Web page are also recommended. As a result, the problems of poor content, simple structure, and lack of candidate keywords for short-text Web pages are solved.

6. CONCLUSION AND FUTURE WORK

Traditional approaches depending on the abundance of textual information in the target Web pages do not work well in the context of short-text Web pages. To address this important problem, we propose a novel approach using the content & advertisement-sensitive PageRank on the Wikipedia graph. In the experiment, our approach yields a high improvement against the traditional approaches in the precision of top 20 keywords on short-text target Web pages. It verifies that content & advertisement-sensitive PageRank is an effective approach to advertising keywords recommendation on short-text Web pages.

In the future work, we plan to refine the Wikipedia graph, such as refining the edges to be more precise to identify the semantic similarity between two entities. For the efficiency improvement of our system, we will implement parallelization. As the WikiGraph has been divided into κ subgraphs based on the category, the computation on these subgraphs can be done in parallel. Furthermore, more parallelism can be achieved using more sophisticated technology of distributed computation for each subgraph. For other applications, we can change the PageRank bias into user profile information and implement our algorithm in personalized search. In addition, our algorithm can be adapted to product recommendation on Web store with the inter-linked product pages.

7. ACKNOWLEDGEMENT

Part of the work is supported by NSF grants IIS-1049694, IIS-1116886, and a Yahoo! Faculty Research and Engagement Grant. Gui-Rong Xue thanks the support by the grants from NSFC project (NO. 60873211) and RGC/NSFC project (NO. 60910123).

REFERENCES

ABHISHEK, V. AND HOSANAGAR, K. 2007. Keyword generation for search engine advertising using semantic similarity between terms. In *Proceedings of the 9th International Conference on Electronic Commerce*. 89–94.

ANAGNOSTOPOULOS, A., BRODER, A., GABRILOVICH, E., JOSIFOVSKI, V., AND RIEDEL, L. 2007. Just-in-time contextual advertising. *CIKM*.

ANTONELLIS, I., GARCIA-MOLINA, H., AND CHANG, C.-C. 2008. Simrank++: Query rewriting through link analysis of the click graph. In *Proceedings of VLDB*. 408–421.

BAEZA-YATED, R., RIBEIRO-NETO, AND B. 2008. *Modem Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc, Boston, MA.

BECKER, H., BRODER, A., GABRILOVICH, E., JOSIFOVSKI, V., AND PANG, B. 2009. What happens after an ad click? quantifying the impact of landing pages in web advertising. In *Proceeding of the 18th ACM conference on Information and knowledge management*. 57–66.

BOLDI, P., BONCHI, F., CASTILLO, C., DONATO, D., GIONIS, A., AND VIGNA, S. 2008. The query-flow graph: model and applications. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*. 609–618.

BOLDI, P., BONCHI, F., CASTILLO, C., DONATO, D., AND VIGNA, S. 2009. Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data*. WSCD '09. 56–63.

BRIN, S., MOTWANI, R., PAGE, L., AND WINOGRAD., T. 1998. What can you do with a web in your pocket. *Bulletin of the IEEE*.

BRIN, S. AND PAGE, L. 1997. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference*. 107–117.

BRODER, A., CICCOLO, P., GABRILOVICH, E., JOSIFOVSKI, V., METZLER, D., RIEDEL, L., AND YUAN, J. 2009. Online expansion of rare queries for sponsored search. In *Proceedings of the 18th international conference on World wide web*.

BRODER, A., FONTOURA, M., JOSIFOVSKI, V., AND RIEDEL, L. 2007. A semantic approach to contextual advertising. In *Proceedings of the 30st annual international ACM SIGIR conference on Research and development in information*. 559–566.

BRODER, A. Z., CICCOLO, P., FONTOURA, M., GABRILOVICH, E., JOSIFOVSKI, V., AND RIEDEL, L. 2008. Search advertising using web relevance feedback. In *Proceeding of the 17th ACM conference on Information and knowledge management*. CIKM '08. New York, NY, USA, 1013–1022.

CAO, H., JIANG, D., PEI, J., HE, Q., LIAO, Z., CHEN, E., AND LI, H. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 875–883.

CARMEL, D., ROITMAN, H., AND ZWERDLING, N. 2009. Enhancing cluster labeling using wikipidia. In *In proceedings of 32nd international ACM SIGIR conference on Research and development in information retrieval*. 139–146.

CHEN, Y., XUE, G., AND YU, Y. 2008. Advertising keyword suggestion based on concept hierarchy. In *Proceedings of the third ACM international conference on Web search and data mining*. 251–260.

CHOI, Y., FONTOURA, M., GABRILOVICH, E., JOSIFOVSKI, V., MEDIANO, M., AND PANG, B. 2010. Using landing pages for sponsored search ad selection. In *Proceedings of the 19th international conference on World wide web*. 251–260.

CRISTO, M., RIBEIRO-NETO1, B., GOLGHER, P. B., AND DE MOURA, E. 2006. Search advertising. In *StudFuzz 197*. 259C285.

FANG, Y., WU, B., LI, Q., BOT, R., AND CHEN, X. 2005. Domain-specific keyphrase extraction. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 283–284.

FENG, J., BHARGAVA, H., AND PENNOCK, D. 2003. Comparison of allocation rules for paid placement advertising in search engines. In *Proceedings of the 5th international conference on Electronic commerce*. 294C299.

HAVELIWALA, T. 2002. Topic-sensitive pagerank. In *Proceedings of the 14th World Wide Web Conference*. 517–526.

HILLARD, D., SCHROEDL, S., MANAVOGLU, E., RAGHAVAN, H., AND LEGGETTER, C. 2010. Improving ad relevance in sponsored search. In *Proceedings of the third ACM international conference on Web search and data mining*. 361–369.

HU, J., FANG, L., CAO, Y., ZENG, H.-J., LI, H., YANG, Q., AND CHEN, Z. 2008. Enhancing text clustering by leveraging wikipedia semantics. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '08. 179–186.

HU, J., WANG, G., LOCHOVSKY, F., SUN, J., AND CHEN, Z. 2009. Understanding user's query intent with wikipedia. In *Proceedings of the 18th World Wide Web Conference*. 471–478.

IAB AND PRICEWATERHOUSECOOPERS. 2011. Available at, http://www.iab.net/media/file/IAB_Full_year_2010_0413_Final.pdf.

JONES, R., REY, B., MADANI, O., AND GREINER, W. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*. 387–396.

JONES, S. AND PAYNTER, G. 2001. Human evaluation of kea, an automatic keyphrasing system. In *Proceedings of the 1th ACM/IEE-CS joint Conference on Digital libraries*. 148–156.

JOSHI, A. AND MOTWANI, R. 2006. Keyword generation for search engine advertising. In *Proceedings of the 6th IEEE International Conference on Data Mining - Workshops*.

KLEINBERG, J. M. 1999. Authoritative sources in a hyperlinked environment. *J. ACM 46*, 604–632.

LITVAK, M. AND LAST, M. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization, Coling 2008*. 17–24.

MATSUO, Y. 2003. Keyword extraction from a single document using word co-occurrence statistical information. In *Proceedings of International Journal on Artificial Intelligence Tools Word Scientific Publishing Company*.

MEDELYAN, O., MILNE, D., LEGG, C., AND WITTEN, I. 2009. Mining meaning from wikipedia. In *International Journal of Human-Computer Studies*. 716–754.

MITRA, M., SINGHAL, A., AND BUCKLEY, C. 1998. Improving automatic query expansion. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 206 – 214.

PAGE, L. 1997. Pagerank: Bringing order to the web. In *Digital Libraries Working Paper*.

PANDEY, S., PUNERA, K., FONTOURA, M., AND JOSIFOVSKI, V. 2010. Estimating advertisability of tail queries for sponsored search. 563–570.

RADLINSKI, F., BRODER, A., CICCOLO, P., GABRILOVICH, E., JOSIFOVSKI, V., AND RIEDEL, L. 2008. Optimizing relevance and revenue in ad search: A query substitution approach. In *proceeding of the 31st international ACM SIGIR conference on Research and development in information retrieval*. 403–410.

RAGHAVAN, H. AND HILLARD, D. 2009. A relevance model based filter for improving ad quality. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '09. 762–763.

RAVI, S., BRODER, A., GABRILOVICH, E., JOSIFOVSKI, V., PANDEY, S., AND PANG, B. 2010. Automatic generation of bid phrases for online advertising. In *Proceedings of the third ACM international conference on Web search and data mining*. 341–350.

RIBEIRO-NETO, B., CRISTO, M., GOLGHER, P., AND MOURA., E. 2005. Impedance coupling in content-targeted advertising. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 496–503.

SCHÖNHOFEN, P. 2006. Identifying document topics using the wikipedia category network. In *Web Intelligence and Agent Systems*. 456–462.

SWENEY, M. 2009. Available at, http://www.guardian.co.uk/media/2009/sep/30/internet-biggest-uk-advertising-sector.

TURNEY, P. D. 2000. Learning algorithms for keyphrase extraction. In *Information Retrieval*. 303–336.

TURNEY, P. D. 2003. Coherent keyphrase extraction via web mining. In *Proceedings of IJCAI'03*. 434–439.

WANG, C., ZHANG, P., CHOI, R., AND EREDITA, M. 2002. Understanding consumers attitude toward advertising. In *Eighth Americas Conference on Informatino System*. 1143–1148.

WANG, H., LING, Y., FU, L., XUE, G., AND YU, Y. 2009. Efficient query expansion for advertisement search. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 51–58.

WANG, P. AND DOMENICONI, C. 2008. Building semantic kernels for text classification using wikipedia. In *proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 713–721.

WANG, P., HU, J., ZENG, H.-J., AND CHEN, Z. 2009. Using wikipedia knowledge to improve text classification. *Knowledge and Information Systems 19*, 265–281.

WEN, J.-R., NIE, J.-Y., AND ZHANG, H.-J. 2001. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*. WWW '01. 162–168.

WITTEN, I., PAYNTER, G., FRANK, E., GUTWIN, C., AND NEVILL-MANNING, C. 1999. Kea: practical automatic keyphrase extraction. In *Proceedings of the 4th ACM conference on Digital libraries*. 254–255.

YIH, W., GOODMAN, J., AND CARVALHO, V. 2006. Finding advertising keywords on web pages. In *Proceedings of the 15th World Wide Web Conference*. 213–222.

YU, J., THOM, J., AND TAM, A. 2007. Ontology evaluation using wikipedia categories for browsing. In *proceeding of the 6th ACM conference on information and knowledge management*. 223–232.

A Brief Survey on RBM and Matrix Factorization for CF

Dingquan Wang

F0703028
5070309650

1 Introduction

In this semester, I mainly focus my laboratory on Restrict Boltzmann Machine and finished a Collaborative Filtering predictor using C++ with the help of Chen Tianqi. Then we try this program on the data set of movielens which is a recommender system and virtual community website that recommends films for its users to watch , based on their film preferences. In the end of April, I participated the KDD Cup 2010 year's Challenge with Zhang Weinan and Zheng Zhao. This year's challenge is to predict student performance on mathematical problems from logs of student interaction with Intelligent Tutoring Systems. This task presents interesting technical challenges, has practical importance, and is scientifically interesting. We reduced this problem to a Collaborative Filtering task, and used the Matrix Factorization approach to solve the problem. This survey is mainly expressing the two topics above I had faced this semester.

2 Restrict Boltzmann Machine

2.1 Boltzmann Machine

A Boltzmann machine is the name given to a type of stochastic neural network, who has an global "Energy" with the units biases and the relative feed in between each units. The "Energy" talked above is defined as follows:

$$E = -\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i \tag{1}$$

In the Boltzmann Machine model, the likelihood of this model can be described as:

$$P(x) \propto exp(\frac{E}{T}) \tag{2}$$

where T is considered to the temperature of this model, indicating the stochastic factor of the model.

In most utilization of the Boltzmann Machine, units of the network are always separated into visible unit v and hidden unit h, the visible unit is always used to

fit the observable facts, and the hidden unit is the higher level feature indicated by the model.Then the "Energy" can be written in the matrix multiplication form as:

$$E(v,h) = -b'v - c'h - h'Wv - v'Uv - h'Vh \tag{3}$$

Since the likelihood is the function of all observed data, then it is the function of the parameter given only by the visible unit. Then we can get the likelihood as follows:

$$P(v) = \sum_h P(v,h) \tag{4}$$

such that:

$$P(v,h) = \frac{exp(-E(v,h))}{Z} \tag{5}$$

For the additional talk, we rewrite the likelihood into log-likelihood:

$$\log P(v,h) = \log \sum_h exp(-E(v,h)) - \log \sum_{v,h} exp(-E(v,h)) \tag{6}$$

We use the gradient ascent to justify the parameters fitting the model, the gradient of single parameter θ can be written as follows:

$$\frac{\partial \log P(v)}{\partial \theta_j} = -\frac{\sum_h exp(-E(v,h))\frac{\partial E(v,h)}{\partial \theta_j}}{\sum_h exp(-E(v,h))} + \frac{\sum_{v,h} exp(-E(v,h))\frac{\partial E(v,h)}{\partial \theta_j}}{\sum_{v,h} exp(-E(v,h))} \tag{7}$$

$$= -\sum_h P(h|v)\frac{\partial E(v,h)}{\partial \theta_j} + \sum_{v,h} P(v,h)\frac{\partial E(v,h)}{\partial \theta_j} \tag{8}$$

$$= - < \frac{\partial E(v,h)}{\partial \theta_j} >_{h|v_i} + < \frac{\partial E(v,h)}{\partial \theta_j} >_{v_i,h} \tag{9}$$

Therefore,from the equation, if sampling from the model was possible, we can obtain a stochastic gradient for use in training the model, as follows. Two samplers are necessary: h given x for the first term, which is called the positive phase, and an (x,h) pair from $P(x,h)$ in what is called the negative phase. A general sampling method is called the Gibbs sampling, which is a special case of MCMC to generate random values from given distribution.

In former Boltzmann Machine, the linkings between different units has no restricted. It can be inferred that the sampling process is of highly cost to making the effective training and predicting process.

In this case the Restrict Boltzmann Machine is proposed by largely reducing the complexity of the former Restrict Boltzmann Machine model. Which has achieve a considerable trade-off between the effectiveness and the accurate of the model.

2.2 Model Description

In the Restrict Boltzmann Machine model, the linkings between the same type of units are extracted, only leaves the edges between visible unit and hidden unit, i.e. the neural network is simplified into a bipartite graph whose visible unit is on one side while the hidden unit on the other. Thus the Energy function can be simplified into

$$E(v,h) = -b'v - c'h - h'Wv \tag{10}$$

where Z is the normalize where $Z = \sum_{x,h} exp(-E(v,h))$ is a normalization term. Where b is the bias of the visible unit, and c is the bias of hidden unit. W is the interaction terms weighting the links between hidden unit and visible unit.

Benefited from this simplification we can keep on refine the equation above to:

$$\frac{\partial \log P(x)}{\partial \theta_j} = -\sum_h P(h|v)\frac{\partial E(v,h)}{\partial \theta_j} + \sum_{v,h} P(v,h)\frac{\partial E(v,h)}{\partial \theta_j} \tag{11}$$

$$= -P(H_i|v)\cdot v_j + E_X[P(H_i = 1|V)\cdot V_j] \tag{12}$$

and by the property of RBM, we can find that:

$$E[H_i|v] = P(H_i = 1|V = v) = \frac{1}{1 + exp(-c_i - W_i x)} = sigm(c_i + W_i x) \tag{13}$$

We can find that we can calculate the expectation precisely and effectively without any sampling process, which can largely speed up the training operation.

For the negative phase we can using an approximation approach called Contrastive Divergence.The $k - step$ Contrastive Divergence involves a second approximation besides the use of $MCMC$ to sample from P. This additional approximation introduces some bias in the initial gradient by starting from the observed data, by this means, the $MCMC$ process can only run a few of k steps.

3 Matrix Factorization for Collaborative Filtering

3.1 Collaborative Filtering

Collaborative Filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets. Collaborative filtering methods have been applied to many different kinds of data including sensing and monitoring data - such as in mineral exploration, environmental sensing over large areas or multiple sensors; financial data - such as financial service institutions that integrate many financial sources; or in electronic commerce and web 2.0 applications where the focus is on user data, etc.

A most simple Collaborative Filtering task is giving a user-item matrix, the predictor try to fill the missing values into the matrix by observing the existing entries. Collaborative filtering systems usually take two steps:

- Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
- Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user

Currently there are mainly two types of approach for Collaborative Filtering:

Memory-Based This mechanism uses user rating data to compute similarity between users or items. This is used for making recommendations. This was the earlier mechanism and is used in many commercial systems. It is easy to implement and is effective. Typical examples of this mechanism are neighborhood based CF and item-based/user-based top-K recommendations.

The advantages with this approach is the explainability of the results, which is an important aspect of recommendation systems. It is easy to create and use. New data can be added easily and incrementally. It need not consider the content of the items being recommended. The mechanism scales well with co-rated items.

Model-Based Models are developed using data mining, machine learning algorithms to find patterns based on training data. These are used to make predictions for real data. There are many model based CF algorithms. These include Bayesian Networks, clustering models, latent semantic models such as singular value decomposition, probabilistic latent semantic analysis, Multiple Multiplicative Factor, Latent Dirichlet allocation, markov decision process based models.

There are several advantages with this paradigm. It handles the sparsity better than memory based ones. This helps with scalability with large data sets. It improves the prediction performance. It gives an intuitive rationale for the recommendations.

3.2 Matrix Factorization

The Matrix Factorization algorithm is a model based approach for Collaborative Filtering, which is focus on finding the latent factor linked both user and item, and making prediction by product the latent factor.

Consider the input is a user-item matrixr_{ui}, we propose to find a joint latent factor space, such that user-item interactions are modeled as inner products in this space. Accordingly, each item i is associated with a vector $q_i \in \mathbb{R}$, and each user u is associated with a vector $p_u \in \mathbb{R}$. For a given item i, the elements off q_i measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product $q_i^T p_u$ captures the interaction between user u and item i the user's overall interest in the items characteristics. This approximates user $u's$ rating of item i, which is denoted by r, leading to the estimate:

$$r_{ui} = q_i^T p_u \tag{14}$$

If we decided the q and p, the ratings can be effectively predicted by above procedure. Now, the major challenge is computing the mapping of each item and user to factor vectors, i.e. to decide q and p, the same method as Restrict Boltzmann Machine, we use the gradient descent method by minimal the loss function as follows:

$$\sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| q_u \|^2) \tag{15}$$

Here λ is to avoid the model parameter overfitting the training data. Now define:

$$e_{ui} = r_{ui} - q_i^T p_u \tag{16}$$

Then the gradient is given as:

$$\nabla q_i = e_{ui} \cdot p_u - \lambda \cdot q_i \tag{17}$$

$$\nabla q_u = e_{ui} \cdot q_i - \lambda \cdot p_u \tag{18}$$

Then the updating law:

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i) \tag{19}$$

$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u) \tag{20}$$

Thus, after several iteration, the loss function may fall into a local minimal area, and the training step is over . The benefit of the Matrix Factorization model is that it is very easy to add features into the loss function and training different models fitting different data.

In this year's KDD Cup with the help of Chen Tianqi, we had implemented the Matrix Factorization algorithm on large scale data set, and try a lot of features such as the problem count, problem id, user history and user profile. And the best result of our result under RMSE is 29.98% (the first place is under RMSE of 27.3%), indicating that the MF of Collaborative Filtering really works and we still have spaces to improve.