Mining Query Logs for Improving Search Efficiency

Shisanu Tongchim[†] **Virach Sornlertlamvanich**[†] **Hitoshi Isahara**[‡]
[†]Thai Computational Linguistics Laboratory
NICT Asia Research Center
112 Paholyothin Road, Klong 1, Klong Luang, Pathumthani 12120, Thailand
[‡]National Institute of Information and Communications Technology
3-5, Hikari-dai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
{shisanu,virach}@tcllab.org, isahara@nict.go.jp

Abstract

The understanding of how users use search queries is an important step towards developing successful web search engines. Mining search query log is a way to gain insight into user behavior. One technique used to analyze the query log is query clustering. Query clustering can be used to group semantically related queries, and can be used to gain an understanding of query usage. In this work, we focus on queries written in Thai language. A technique for query clustering is proposed. We first discuss the nature of Thai queries that makes this problem much more challenging. We examine the use of returned results from several search engines to enrich the meanings of queries. The experimental results show that query enrichment helps in finding the degree of similarity between any two queries. We apply a density-based clustering technique, called DBSCAN, for the query clustering task. Finally, we explore the use of clustered queries for query expansion.

Keywords: Query Clustering, Query Similarity

1 Introduction

Search engine query logs are valuable sources of information regarding the interactions between users and search providers. Data mining has been applied on query logs in order to gain some insights into user behavior and to improve the search performance. Query clustering is one of major research areas conducting on query logs. The objective is to group semantically related queries. The results can be applied in several ways. They can be used to analyze the query usage. For example, the most frequently asked questions or the current trends of increasingly asked questions can be identified. If we know this information, the search performance for these questions can be improved. Another application is the query recommendation task. Some related queries to user queries are suggested to users in order to refine their queries if the results of the current queries are unsatisfying.

In this work, we carry out some experiments in clustering search queries. We are particularly interested in the queries written in Thai language. Firstly, we discuss some challenging issues in clustering the Thai query log that we have. User queries are generally short and lack of information about user intention. Many of them also contain typos and ill-written strings. The use of words from queries is thus insufficient in determining the similarity degree between any two queries. In our case, there are also some language-related issues, e.g. unknown word problem, that may affect the performance in finding the query similarity. Before conducting query clustering, we investigate some similarity functions based on the use of search engine results to enhance the similarity calculation. Then, we explore the use of a density-based approach, called DBSCAN, for clustering search queries. After clustering queries, we examine the use of clustered results for query expansion.

2 Related Work

Baeza-Yates *et al.* [1] used a simple K-means algorithm for query clustering. They used extracted words from selected URLs by users for determining query similarity. However, the use of K-means algorithm requires the number of clusters as an input parameter which is impractical to determine in advance.

Chuang and Chien [2] used hierarchical ag-

glomerative clustering algorithm (HAC) for query clustering. However, the input data for query clustering task usually contains many outliers. The domains with many outliers are undesirable for hierarchical agglomerative clustering [3; 4].

Beeferman and Berger [5] exploited clickthrough data for query clustering. They used a graph-based iterative clustering based on the information of user queries and selected URLs. However, it is hard to justify a promising termination condition. Chan *et al.* [6] adopted the technique of Beeferman and Berger for clustering search queries. They modified the termination condition since the original termination condition grouped the majority of queries into a single cluster.

Wen *et al.* [4; 3] used DBSCAN for clustering search queries. The similarity calculation was based from several features (e.g., query content words, clickthroughs). Their experiments tried to examine the effect of each feature combination for the clustering results.

3 Thai Query Log

The queries used in this study are obtained from Truehits[1]. Truehits is a largest web statistics collector in Thailand. Truehits provides the list of alphabetically sorted queries submitted to search engines. To our knowledge, this is the only public source of used Thai queries. However, there are two challenging issues in clustering this Thai query log. The first one is a lack of useful information that can be used to determine the similarity degree between any two queries. The second issue is related to the language aspects.

3.1 Limited information

Typically, queries submitted to search engines are short and ambiguous. Many queries are ill-written or misspelled. The use of words from queries are thus inefficient for finding the query similarity and clustering the related queries. Many studies have utilized other information in finding the similarity or clustering user queries. These studies use some sorts of human activities recorded by search engines. Some studies [6; 1] rely on the information about the documents clicked by users for the corresponding queries (called *clickthroughs*). The assumption is that two queries are similar or closely related if both queries lead to the same set of documents selected by users. Some proposed techniques [7; 8] try to identify the set of queries issued by a user within a certain time frame. These studies try to identify the user sessions from the search engine logs. The intuition behind these techniques is that a user submits several related queries during searching for information.

The studies mentioned earlier require the query log with some information of user activities. However, such information is not always publicly available. Although most search engines record this information in order to improve their service, the disclosure of such information may lead to privacy concern, like the case of AOL [9; 10]. In our case, Truehits only provides the list of used queries from various search engines. These is a lack of useful information that can be used to determine the similarity degree of these Thai queries. Thus, the techniques mentioned earlier cannot be applied to our case. To overcome the problem of short and ambiguous queries, a technique called *query enrichment* has been examined. By submitting queries to search engines, some results can be obtained. The textual information from these returned results is used to resolve and enrich the meanings of queries. That is, search engines are used as a tool to resolve the meanings of queries.

3.2 Language aspects

Despite a lack of information for query clustering, the nature of Thai queries is also challenging for this task. Thai language is a non-segmenting language. Thai text is written without explicit word boundary. Therefore, the Thai text processing normally starts from applying the word segmentation. Due to the ambiguity of word boundary, the errors of word segmentation are generally unavoidable. In case of search queries, many keywords are ill-written or misspelled. This makes unreliable results when applying the word segmentation. The inevitable errors of the word segmentation clearly affect the similarity function and query clustering in the later steps.

From the observation of query log, there are

[1] truehits.net

two main groups of queries based on the query formation. In the first group, each query is composed of some manually selected words. The keywords are separated with spaces like English queries. In the second group, each query is composed of one or few short phrases. In this group, the word segmentation plays an important role in extracting words from the query strings for processing in the later steps. The correctness of word segmentation clearly influences the performance of the query similarity calculation. Based on the query log analysis, there are several issues that would degrade the performance of word segmentation and may affect the query similarity calculation.

- **Unknown word problem**

  New words regularly appear throughout the Web. Since many word segmentation algorithms rely on dictionaries, the accuracy and completeness of dictionaries play an important role in the correctness of word segmentation. For example, the query "ก้านกล้วย" (Kan Kluai) which is the name of a Thai animation will separated into two words, "ก้าน" (limb) and "กล้วย" (banana), by the word segmentation. When comparing this query with others, it may be confused by some queries containing the word "กล้วย" (banana).

- **Improper query usage**

  Improper queries are typically found in the query log. They contains typos, ill-written strings or informal language. For example, a query found in the query log is "กรมอุตุนิยมวิทยา" which is the "Thai Meteorological Department". This query can be considered as two words: "กรม" (department) and "อุตุนิยมวิทยา" (meteorology). Since this word represents a unique entity, it may be recognized as an indivisible unit. We have found that there are some queries that resemble to this word, but they are ill-written. For example, we found at least three queries that can be considered to refer to this word: "กรมอุตุ", "กรมอุตุนิยม" and "กรมอุตุวิทยา". The use of these keywords usually leads to the websites that have improper forms of the word "กรมอุตุนิยมวิทยา", rather than the website of the Thai Meteorological Department.

  When applying the word segmentation to these queries, the results may be unexpected. For example, the query "กรมอุตุนิยม" will be separated into three words: "กรม" (department), "อุตุ" (comfort) and "นิยม" (favour). Thus, it is difficult when trying to relate this query to "กรมอุตุนิยมวิทยา".

- **Incorrect or inconsistency transliteration**

  There are many transliterated words in the list of queries. They are foreign words written by using Thai characters. One problem arises when each user has a different way for transliteration. For example, the word "Internet" has been transliterated into several forms: "อินเตอร์เน็ต", "อินเทอร์เน็ต" and "อินเตอร์เน็ท". This issue affects the similarity measurement among queries that have different forms of transliteration. Moreover, the dictionary or lexicon used by the word segmentation algorithm may not have all forms of transliterated words. In this case, the unknown word problem may occur.

4 Query Similarity

Before applying query clustering, the similarity function must be defined. To overcome the problem of short and ambiguous queries, the results from search engines are used to resolve and expand the meanings of queries. This technique is called query enrichment. A similar approach to this technique was used by the team who won ACM KDDCUP in 2005 [11]. Their problem is to classify given queries to predefined categories.

In this study, we try to use this technique to overcome the problem of limited information and the language aspects mentioned earlier. The process of query enrichment is as follows:

1. Each query is submitted to a number of public web search engines.

2. The titles and short descriptions (snippets) are parsed from the returned results. The word segmentation is applied. The extracted words from the query string are also

merged into the list of extracted words from the titles and snippets. Stopwords and special symbols are filtered out. Stemming is not applied since Thai language does not have the concept of inflection.

3. We filter out the words that have low frequency.

4. The extracted words are then weighted by using *tf*idf*.

We examine the similarity functions on the query recommendation task. From 45,792 Thai queries obtained from Truehits, we select 9,000 queries for evaluating the similarity functions. We select 60 queries as the main topics. The task is to find the top 30 queries that are likely to be relevant to each given main topic. We adopt the mean average precision (MAP) as the evaluation measure. The suggested results are evaluated by a team of three judges.

Query enrichment is based on the results from three search engines: Google[2], SiamGURU[3] and AlltheWeb[4]. The top 20 results from each search engine are used. The word segmentation is based on the maximal matching method.

The cosine similarity is used to compute the similarity score between a pair of queries. Three similarity functions are tested:

1. *Query words only* : This function uses words extracted from queries without query enrichment. This test is used as a baseline for assessing the benefit of the proposed method. However, using *tf*idf* for the query words is impractical. Most query words appear one time only in each query. Thus, *tf* for the query words makes no sense. Some studies suggest the use of the search frequency instead of the term frequency [12]. However, Truehits does not provide the information about the search frequency. Thus, we use *idf* to weight the extracted terms in the experiment on the query words.

2. *Query enrichment* : This function uses query enrichment as mentioned earlier. Each term is weighted by using *tf*idf*.

3. *Hybrid method* : We also examine a hybrid method by combining the similarity scores from the use of query enrichment and the similarity of query words. A simple combination is used as follows:

$$sim_{hybrid} = \alpha \times sim_{enrich} + \beta \times sim_{query} \quad (1)$$

where sim_{enrich} and sim_{query} are the cosine similarity scores with and without query enrichment respectively. To select the best coefficients, we fine-tune the parameters by using a small set of 6 topics apart from the topics used in the main evaluation. In the experiment, we set $\alpha = 0.9, \beta = 0.1$.

Table 1: MAP for the query recommendation task

	Method	MAP
1	Query Words	.560
2	Query Enrichment	.693
3	Hybrid	.711

The results of three variants of similarity calculation are shown in Table 1. The results show that the use of query enrichment helps in improving the similarity calculation. From the experiment, the best similarity function is the hybrid function.

5 Query Clustering

After choosing the similarity function, the next step is to determine the a clustering algorithm. A density-based approach, called *DBSCAN* [13], is used in the query clustering. DBSCAN has several advantages for this task. It does not require some parameters that are impractical to determine in advance (e.g., the number of clusters). This technique is also suitable of the data with outliers. In the query clustering task, not all queries can be clustered. These queries are unique and are not related to any other queries. Thus, these queries should not be classified as

[2]www.google.com
[3]www.siamguru.com
[4]www.alltheweb.com

members of any clusters. DBSCAN is suitable for this kind of data. DBSCAN can identify these queries as outliers.

DBSCAN is a single pass algorithm. Each point in one cluster is *density-reachable* from other points in the same cluster. The algorithm groups all points that are density-reachable from the starting point to the same cluster. DBSCAN requires two parameters, MinPts and Eps, to control the notion of density. MinPts is the minimum number of points while Eps is the distance threshold. The clustering result depends upon how both parameters are set. If the density is too high, only few clusters will be detected. In contrast, several clusters will be indivisible from each other if the density is too low.

To examine how the parameters affect the clustering results, some experiments are carried out by using different parameter settings for query clustering. The clustering results are manually checked. We adopt two standard performance measures, precision and recall, to assess the quality of clustering results. Precision can be judged by the ratio of the numbers of correctly clustered queries to the size of clusters. However, recall is difficult to determine since it requires the information about all correct members to each cluster. Some studies [3; 4] use normalized recall instead of standard recall. The normalized recall is calculated relative to the clustering result of the best algorithm. However, the normalized recall shows only the performance of algorithms when compared against the best one. It does not provide real recall information. In order to do the quantitative assessment in terms of precision and recall while the performance evaluation is still manageable, we use a limited number of queries in the experiments. We use a set of 9,000 queries from the previous section to test the effect of parameter settings. The finding will be applied to the later experiment.

Three parameter settings are examined. For the sake of simplicity, MinPts is set to a constant value of 3. Eps is set as 1.3, 1.5 and 1.7 for three experiments. Intuitively, the use of lower density by increasing Eps should improve recall, but may reduce precision. In contrast, higher density by reducing Eps seems to improve precision at the expense of recall. We evaluate the performance of each parameter setting on 20 randomly chosen clusters. The performance of three parameter settings is shown in Table 2. From the table, the results confirm the intuition of precision and recall tradeoff. When reducing the density by setting $Eps = 1.7$, recall improves at the expense of precision. When increasing the density by setting $Eps = 1.3$, precision improves nearly to the maximum value. We also calculate the F_1 measure for each parameter setting. The F_1 measure represents the overall performance. From the F_1 scores, the setting $MinPts = 3, Eps = 1.5$ yields the best overall performance among three settings.

Table 2: Precision and recall for three parameter settings

Parameter setting	Pr	Re	F_1
$MinPts = 3$, $Eps = 1.7$	0.851	0.804	0.827
$MinPts = 3$, $Eps = 1.5$	0.927	0.754	0.832
$MinPts = 3$, $Eps = 1.3$	0.984	0.608	0.752

5.1 Multi-level clustering

We apply the best parameter setting ($MinPts = 3, Eps = 1.5$) from the previous section to cluster a larger set of Thai queries from Truehits (45,792 queries). The clustering results show that 1163 clusters can be detected. These clusters are composing of 14,531 queries. Thus, the size of outliers is 31,261. However, we observe that there are some clusters that compose of two or more sub-clusters. These clusters can be further divided into several clusters. For example, the cluster number two contains the queries related to "กระเป๋า" (Bag) and "กล้อง" (Camera). The clustering algorithm merges these two clusters into the same cluster since there are some queries that bridge the gap between two clusters and make them density-reachable, e.g. the query 1223 "กระเป๋ากล้อง" (Camera bag). This situation can be solved by increasing the density, but recall may decrease as shown in the previous section.

To obtain indivisible clusters without the penalty of reducing recall, we propose a variant of DBSCAN called multi-level DBSCAN. In

multi-level DBSCAN, a recursive DBSCAN is carried out. From the output of DBSCAN, each cluster is partitioned by a recursive call of DBSCAN using higher density level. The process can be repeated recursively on existing clusters until all clusters cannot be further decomposed. By using higher density level to decompose existing clusters, three conditions can arise:

- *Condition 1 :* All queries are considered as outliers.
- *Condition 2 :* Only one cluster can be found. Some queries are considered as outliers.
- *Condition 3 :* Two or more sub-clusters can be found. Some outliers are detected.

In the conditions 1 and 2, we assume that the cluster is indivisible and all queries belong to this cluster. We use these two conditions as stopping criteria. In the condition 3, the cluster can be divided into several clusters. The outliers are assigned to the nearest clusters. A recursive DBSCAN is then applied to these clusters until the conditions 1 or 2 are met.

The incremental density of multi-level DBSCAN is used to decompose the clusters containing some sub-clusters. The assignment of outliers to nearest clusters helps in retaining recall when the density level increases. In order to evaluate the precision of outlier assignment, we apply multi-level DBSCAN on the set of 45,792 Thai queries. Like the previous test, MinPts is set to a constant value of 3. The Eps is set to 1.5 as the starting value with a decreasing step size of 0.2. At the first level, the clustering results are identical to the use of DBSCAN with $Eps = 1.5$. Then, the clustering algorithm is repeated with higher density levels ($Eps = 1.3, 1.1, ...$). From the experiment, multi-level DBSCAN mostly terminates at the second level ($Eps = 1.3$), while some clusters can be further divided until the third level ($Eps = 1.1$). We randomly choose 30 clusters for evaluating the outlier assignment. From these 30 clusters, 197 outliers are assigned to one of these clusters. If outliers correspond with existing members in assigned clusters, they will be judged as correct assignments. In contrast, incorrect assignments occur when outliers should assign to other clusters instead of the current clusters or should be discarded. By manually evaluating, the average precision of the outlier assignment is 0.8.

Table 3: Precision and relative recall of DBSCAN ($Eps = 1.3$) and multi-level DBSCAN

Parameter setting	Pr	RR
DBSCAN ($Eps = 1.3$)	0.94	0.65
Multi-level DBSCAN	0.87	1

We compare the results from DBSCAN with the high density level ($Eps = 1.3$) and multi-level DBSCAN. This test is used to compare the results from multi-level DBSCAN and the use of DBSCAN with high density level in ability to distinguish close clusters. Since it is impractical to determine recall for the large dataset, we use relative recall calculated by normalizing recall of each method with recall of multi-level DBSCAN. Relative recall shows the recall metric of each method relative to multi-level DBSCAN. Basically, relative recall is calculated by dividing the number of correctly clustered queries of each method with the number of correctly clustered queries of multi-level DBSCAN. We use 20 clusters for evaluation. The results are shown in Table 3. The results indicate that precision of multi-level DBSCAN is slightly less than that of DBSCAN, while recall is much better. The reducing in precision is caused by errors in the outlier assignment. Overall, multi-level DBSCAN can be used to divide large clusters containing several sub-clusters without sacrificing recall. It is also applicable when the optimal parameter setting is not known.

6 Automatic Query Expansion

The results from the previous sections can be used to determine some statistics about user queries, e.g. the most frequently asked questions, the diversity of query formations in asking the same question. In addition, the results can be applied for the query suggestion task as shown in the section 4. In this section, we try to use the results of query clustering for the query expansion. The idea is to use some related queries to expand the search results of user queries. When a user submits a query to the system, the system

automatically finds some related queries to the user query and submits the user query and related queries to a search engine. The search results are then merged by using some meta-search approaches.

We simulate this idea by randomly selecting 20 topics from the query log as the main topics. For each main topic, three related queries are selected as support queries by using the queries from the same cluster of the main topic. The main topics and their support queries are submitted to Google, and the search results are collected. The results from each main topic and its support queries are merged by using Borda-fuse [14]. We assign the same weight to the results from each query.

Like the evaluation of the similarity functions, the performance measure is MAP. The binary judgement is used. Two systems are compared. The first one uses only the results from the main topics. The top 20 returned results of each main topic are evaluated. In the second system, query expansion is applied. The results from each main topic and support queries are merged into a single ranked result by using Borda-fuse. By submitting each main topic and its support queries to the search engine, the top 20 returned results of each query is collected and merged into a single pool. Each main topic has three support queries. Thus, the maximum pool size of returned results for each main topic and support queries is 80. We apply Borda-fuse to select the top 20 results from the result pool.

We develop a web-based user interface for the blind evaluation. The results from two systems are merged and shown randomly on the interface. We employ two human assessors to evaluate the results. Based on the results from two systems on 20 topics, there are 229 relevant results from the total 800 results. The first system can find 139 relevant results, while the second system that use query expansion can find 161 relevant results. In terms of MAP, the first system has the MAP score as .37, while MAP of the second system that uses query expansion is .39. From the results, there is no obvious improvement from query expansion. A possible explanation of this finding is the sparseness of results from the main topic and its support queries. Borda-fuse works by calculating the Borda score of each result. Given one main topic and three support queries, we normally get four lists of returned results. Each result gets a number of points, depending on the positions in these lists. The higher ranked results have the higher scores. The total point of each result is determined from the positions in these four result lists. The prospective results will be the results ranked in the high positions in several lists.

Based on our previous study [15] in applying meta-search models on the results from several search engines, the documents have a higher chance of being relevant if they can be found by several search engines. We analyze the result lists from each main topic and its support queries. Despite the similarity of the main topics and support queries, the results have a low degree of overlap. The majority of results (90.25%) are unique to one of the queries. The results differ from queries to queries. The sparseness of results and a lack of agreement between the results from the main topic and the support queries may not work well with Borda-fuse. Therefore, the retrieval effectiveness cannot be boosted by the fusion model. However, we acknowledge that some extensive experiments are necessary to clarify the clause of this issue.

7 Conclusions

The analysis of search query log can be used to gain some insight user behaviour. The findings can be used to improve the search engine performance to match user demands. In this work, we study some techniques for mining search query logs. We start from designing the similarity functions that can be used when the information is really limited. We show that the similarity functions can be applied to the query suggestion task. Then, we investigate some techniques for query clustering. The clustered results provide some important statistics that can be used to improve the search performance. Finally, we try to use the clustered results for query expansion. The experimental results show that the improvement is marginal. One possible explanation is the sparseness of results from multiple queries.

References

[1] Ricardo A. Baeza-Yates, Carlos A. Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In Wolfgang Lindner, Marco Mesiti, Can Türker, Yannis Tzitzikas, and Athena Vakali, editors, *EDBT Workshops*, volume 3268 of *Lecture Notes in Computer Science*, pages 588–596. Springer, 2004.

[2] Shui-Lung Chuang and Lee-Feng Chien. Towards automatic generation of query taxonomy: A hierarchical query clustering approach. In *ICDM*, pages 75–82. IEEE Computer Society, 2002.

[3] Ji-Rong Wen and HongJiang Zhang. Query clustering in the web context. In Weili Wu, Hui Xiong, and Shashi Shekhar, editors, *Clustering and Information Retrieval*, pages 195–226. Kluwer, 2003.

[4] Ji-Rong Wen, Jian-Yun Nie, and HongJiang Zhang. Query clustering using user logs. *ACM Trans. Inf. Syst.*, 20(1):59–81, 2002.

[5] Doug Beeferman and Adam L. Berger. Agglomerative clustering of a search engine query log. In *KDD*, pages 407–416, 2000.

[6] Wing Shun Chan, Wai Ting Leung, and Dik Lun Lee. Clustering search engine query log containing noisy clickthroughs. In *SAINT*, pages 305–308. IEEE Computer Society, 2004.

[7] Bruno M. Fonseca, Paulo Braz Golgher, Edleno Silva de Moura, Bruno Pôssas, and Nivio Ziviani. Discovering search engine related queries using association rules. *J. Web Eng.*, 2(4):215–227, 2004.

[8] Bruno M. Fonseca, Paulo Braz Golgher, Edleno Silva de Moura, and Nivio Ziviani. Using association rules to discover search engines related queries. In *LA-WEB*, pages 66–71. IEEE Computer Society, 2003.

[9] Electronic Frontier Foundation. AOL's massive data leak. http://www.eff.org/Privacy/AOL/, 2007. last accessd on May 18, 2007.

[10] Michael Barbaro and Tom Zeller Jr. A face is exposed for AOL searcher no. 4417749. New York Times, August 9 2006.

[11] Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. Query enrichment for web-query classification. *ACM Trans. Inf. Syst.*, 24(3):320–352, 2006.

[12] Zhiyong Zhang and Olfa Nasraoui. Mining search engine query logs for query recommendation. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *WWW*, pages 1039–1040. ACM, 2006.

[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[14] Javed A. Aslam and Mark H. Montague. Models for metasearch. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *SIGIR*, pages 275–284. ACM, 2001.

[15] Shisanu Tongchim, Virach Sornlertlamvanich, and Hitoshi Isahara. Examining the feasibility of metasearch based on results of human judgements on thai queries. In *Proceedings of The 2007 IEEE International Symposium on Data Mining and Information Retrieval*, 2007.