Subtopic Mining via Modifier Graph Clustering

Hai-Tao Yu and Fuji Ren

The University of Tokushima, Japan
yu-haitao@iss.tokushima-u.ac.jp,
ren@is.tokushima-u.ac.jp

Abstract. Understanding the information need encoded in a user query has long been regarded as a crucial step of effective information retrieval. In this paper, we focus on *subtopic mining* that aims at generating a ranked list of subtopic strings for a given topic. We propose the *modifier graph* based approach, under which the problem of subtopic mining reduces to that of graph clustering over the modifier graph. Compared with the existing methods, the experimental results show that our modifier-graph based approaches are robust to the sparseness problem. In particular, our approaches that perform subtopic mining at a fine-grained term-level outperform the baseline methods that perform subtopic mining at a whole query-level in terms of *I-rec*, *D-nDCG* and *D#-nDCG*.

Keywords: Subtopic mining, Kernel-object, Modifier graph.

1 Introduction

Recent studies show that billions of daily searches are made by web users. Instead of formulating natural language queries, the vast majority of users are submitting short queries with little or no context, which are often ambiguous and/or under-specified. For example, the query *Harry Potter* could refer to a *book* or a *movie*. For the *movie*, a user may be interested in the *main character* or the *reviews*. In view of the above-mentioned facts, the technique of *search result diversification* (e.g., [1, 13]) has been proposed and attracted significant attention. It provides a diversified search result, which features a trade-off between *relevance* (ranking more relevant web pages in higher positions) and *diversity* (satisfying users with different information needs). For a better diversified search result, an accurate estimation of the information needs or subtopics underlying a given topic, which is referred to as *subtopic mining* [17, 19], becomes an important problem. Since the query log records the historical search behaviors performed by massive users (e.g., clicked web pages in response to a query, subsequently reformulated queries, etc), many researchers [10, 14, 16] performed subtopic mining via query log mining and have shown that the click information (e.g., page co-click and session co-occurrence) are valuable for determining the topical relevance of queries. However, as shown by Bonchi et al. [5], many methods have been proposed for and tested on head queries, which work poorly for unseen queries or queries with sparse click information. As query frequency is known to obey the power-law, these methods leave a large ratio of queries uncovered.

V.S. Tseng et al. (Eds.): PAKDD 2014, Part I, LNAI 8443, pp. 337–347, 2014.

In this paper, we propose an effective approach that performs subtopic mining at the term level, which outperforms baseline methods that work at the query-level. Subsequently, Section 2 discusses the related work. Section 3 formalizes the target problem. Section 4 details the proposed approach. Section 5 discusses the experimental results. We conclude our work in section 6.

2 Related Work

To capture the underlying subtopics encoded within user queries, considerable studies [6, 7, 15, 21, 22, 23] have been conducted from various aspects. For example, Beeferman and Berger [2] viewed the query log as a bipartite graph, and the obtained clusters were interpreted as subtopics covering diverse queries. The research by Jones and Klinkner [11] showed that users' search tasks are interleaved or hierarchically organized. They studied how to segment sequences of user queries into a hierarchical structure. Sadikov et al. [16] and Radlinski et al. [14] tried to infer the underlying subtopics of a query by clustering its refinements. The web page co-click and session co-occurrence information were viewed as indicators of topical relevance. Hu et al. [10] interpreted a subtopic as a set of keywords and URLs, their key intuition was that: many users add one or more additional keywords to expand the query to clarify their search intents.

Different from the above methods that have treated a whole query as the minimum analysis unit. A number of studies (e.g., [20, 21]) performed intent analysis at a term-level. For example, Wang et al. [21] studied how to extract broad aspects from query reformulations, each broad aspect is represented by a set of keywords. Rather than raw terms based subtopic analysis, we perform subtopic mining by encapsulating intent roles [24]. We believe that our approach achieves a robust subtopic mining, especially for tail or unseen queries.

3 Problem Formalization

The *Intent* tasks [17, 19] of *NTCIR-9*[1] and *NTCIR-10*[2] are the prototypes on which our formalization builds. The core notations are: (1) **Topic ($\ddot{T}$)**: it refers to an input instance for testing; (2) **Subtopic ($\ddot{t}$)**: it refers to a possible information need or an intent underlying a topic; (3) **Subtopic string ($tStr$)**: it is viewed as an expression of a subtopic. For example, for the topic *Harry Potter*, *Harry Potter fiction* and *Harry Potter reading* are two subtopic strings about the subtopic *book*. A topic and a subtopic string can either be a real query or a query-like string derived from other resources, e.g., query suggestions.

Formally, for a given topic $\ddot{T}$, suppose there are k possible subtopics $X = \{\ddot{t}_1, ..., \ddot{t}_k\}$. Let $Y = \{tStr_1, ..., tStr_n\}$ be a set of n subtopic strings with respect to X. The problem of subtopic mining is formalized as: finding a ranked list of subtopic strings L that ranks subtopic strings representing more popular

[1] `http://research.nii.ac.jp/ntcir/ntcir-9/index.html`
[2] `http://research.nii.ac.jp/ntcir/ntcir-10/index.html`

subtopics at higher positions and includes as many subtopics as possible. The quality of L depends on its consistency with the ideal ranked list L^* (e.g., a ranked list created by human assessors).

4 Subtopic Mining through Modifier Graph Clustering

4.1 Intent Role Oriented Modifier Graph

Let Γ, Q, s, q, t, d denote the query log, query set, a session, a query, a term and a web page respectively, $D(q) = \{d\}$ denote the clicked web pages of q, $t \succ q$ denote that t is a composing term of q, $q \in s$ denote that q occurred in s. Fig. 1 shows some click behaviors, where q_1 and q_2 occur in session s_k, d_1, d_2 and d_3 are the clicked web pages. Some query log oriented notations are given as: (1) **Co-Session**: For $q_i \neq q_j$, iff $\exists s$ that meets $q_i \in s$ and $q_j \in s$, q_i and q_j are co-session queries, i.e., $CoSession(q_i, q_j)$, e.g., q_1 and q_2 in Fig. 1; (2) **Co-Click**: For $q_i \neq q_j$, iff $D(q_i) \cap D(q_j) \neq \emptyset$, q_i and q_j are co-click queries, i.e., $CoClick(q_i, q_j)$, e.g., q_1 and q_2 in Fig. 1; (3) **Co-Query**: For $t_i \neq t_j$, iff $\exists q$ that meets $t_i \succ q$ and $t_j \succ q$, t_i and t_j are co-query terms, i.e., $CoQuery(t_i, t_j)$, e.g., t_2 and t_3 in Fig. 1; (4) **Term-Level Co-Session**: For $t_i \neq t_j$, iff $\exists q_m \in Q$ and $\exists q_n \in Q$ that meet $CoSession(q_m, q_n) \wedge t_i \succ q_m \wedge t_j \succ q_n$, t_i and t_j are co-session terms, i.e., $TCoSession(t_i, t_j)$, e.g., t_1 and t_2 in Fig. 1; (5) **Term-Level Co-Click**: For $t_i \neq t_j$, iff $\exists q_m \in Q$ and $\exists q_n \in Q$ that meet $CoClick(q_m, q_n) \wedge t_i \succ q_m \wedge t_j \succ q_n$, t_i and t_j are co-click terms, i.e., $TCoClick(t_i, t_j)$, e.g., t_1 and t_3 in Fig. 1. In this paper, the statistics of co-session and co-click of a query log that are defined at a query level, are called *query-level knowledge in query log* (QKQL). In contrast, the statistics of co-query, term-level co-session and term-level co-click that are defined at a term level are called *term-level knowledge in query log* (TKQL).

Rather than raw terms based analysis, we encapsulate the terms with intent roles (*Kernel-object* & *modifier*) by Yu and Ren [24]. Kernel-object (*ko*) refers to the term that abstracts the core object of the underlying subtopic encoded in a query. Modifier (*mo*) refers to the co-appearing terms with kernel-object, which explicitly specify user's interested aspects. A query that can be represented with kernel-object and modifier is defined as a *role-explicit* query. Otherwise, a *role-implicit* query. Because topic and subtopic string are query-like strings, thus, they can be analogously classified as role-explicit ones and role-implicit ones. Table 1 shows 3 role-explicit subtopic strings, the 1st column (*In/Out*) implies whether they are included in query log SogouQ(Section 5.1). For convenience, we directly use kernel-object and modifier to refer to the terms annotated as kernel-object and modifier respectively. Moreover, when determining the kernel-object and modifier for a given role-explicit query, topic or subtopic string, the method by Yu and Ren [24] is used as a black box in our study, which selects the annotation with the maximum likelihood as the optimal annotation using a generative model.

Definition 1 (Co-kernel-object Elements). *Given a kernel-object ko, co-kernel-object elements refer to a set of role-explicit subtopic strings that share the same kernel-object, denoted as CoKO(ko)=*$\{tStr\}$.

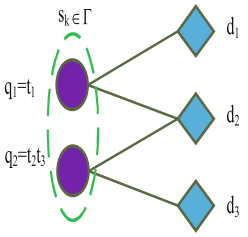

Fig. 1. Example click behaviors

Table 1. Role-explicit subtopic strings

In/Out	Annotated subtopic string
In	哈利波特游戏(*Harry Potter game*) *ko mo ko mo*
In	哈利波特小说(*Harry Potter fiction*) *ko mo ko mo*
Out	哈利波特阅读(*Harry Potter reading*) *ko mo ko mo*

Definition 2 (Modifier Graph). *Modifier graph* $G_{ko} = \{V, E\}$ *is an undirected weighted graph derived from co-kernel-object elements* $CoKO(ko)$*, (i) The node set consists of modifiers, i.e.,* $V = \{mo|tStr \in CoKO(ko), mo \succ tStr\}$*; (ii) The edge set is given as:* $E = \{e = (mo_i, mo_j)|\delta_{TKQL}(mo_i, mo_j) > 0\}$*, where* $\delta_{TKQL}(mo_i, mo_j) = \lambda_1 \frac{CoQuery(mo_i,mo_j)}{max_V CoQuery+1} + \lambda_2 \frac{TCoSession(mo_i,mo_j)}{max_V TCoSession+1} + \lambda_3 \frac{TCoClick(mo_i,mo_j)}{max_V TCoClick+1}$*, namely, the TKQL of a pair of modifiers is normalized by the maximum ones (*$max_V CoQuery$*,* $max_V TCoSession$ *and* $max_V TCoClick$*) respectively, and the edge weight is a linear combination of the normalized values.*

Assuming that $CoKO(ko = 哈利波特(HarryPotter))$ merely consists of the three subtopic strings in Table 1, Fig. 2 shows the corresponding modifier graph based on the TKQL of SogouQ in Table 2(given as *CoQuery:TCoSession:TCoClick*). Obviously, 小说(fiction) and 阅读(reading) are strongly interacted (a weight of 0.9991); 游戏(game) and 小说(fiction) (a weight of 0.0313), 游戏(game) and 阅读(reading) (a weight of 0.0049) are weakly interacted. If we perform graph clustering over this modifier graph, 游戏(game) is likely to be grouped into a cluster. 小说(fiction) and 阅读(reading) are likely to be grouped into another cluster. Many studies (e.g., [4, 5, 14, 22]) have shown that the QKQL helps to determine the topical relevance among queries. Analogously, the TKQL derived from QKQL manifests the topical relevance among terms. Put another way, the larger value of $\delta_{TKQL}(mo_i, mo_j)$, the more relevant subtopics indicated by mo_i and mo_j. Thus, 小说(fiction) and 阅读(reading) express a relevant subtopic, 游戏(game) expresses a different subtopic. Because the subtopics underlying the co-kernel-object elements depend on the composing modifiers, we can further deduce that 哈利波特阅读(Harry Potter reading) and 哈利波特小说(Harry Potter fiction) express the same or similar subtopic, while 哈利波特游戏(Harry Potter game) express a different subtopic. Not surprisingly, the partition of a modifier graph uncovers the prior unknown subtopics of a set of co-kernel-object elements.

4.2 Modifier Graph Construction and Clustering

Given the kernel-object ko with respect to a topic $\ddot{T}$, we construct the modifier graph G_{ko} through the following two steps: **Step-1**: Obtaining sufficient

Table 2. TKQL derived from SogouQ

	游戏(game)	小说(fiction)	阅读(reading)
游戏(game)	×	7:78:20	1:13:1
小说(fiction)	7:78:20	×	700:993:3637
阅读(reading)	1:13:1	700:993:3637	×

Fig. 2. A modifier graph

co-kernel-object elements $CoKO(ko)$. We first manage to obtain a set of candidate subtopic strings, say, $\{ctStr\}$, which is further used as the base for generating $CoKO(ko)$. Here *recall* is the key point, it ensures the coverage of different subtopics. The adopted resources are: (1) *Query log*; (2) *Query suggestions* for topic $\ddot{T}$; (3) *Noun phrase (NP) and verb phrase (VP) segments* extracted from the result snippets for topic $\ddot{T}$ by a search engine. Specifically, all the queries or segments that include ko as a substring are selected as candidate subtopic strings. The query suggestions are directly selected as candidate subtopic strings. According to Definition 1, the co-kernel-object elements are all role-explicit ones that can be represented with kernel-object and modifiers. When deriving the $CoKO(ko)$ based on a set of candidate subtopic strings, we relax the restriction as: regardless of whether a candidate subtopic string is role-explicit or role-implicit, once it includes ko as a substring, we intuitively assume that it is a co-kernel-object element. The substring ko is directly regarded as its kernel-object, the remaining parts are directly segmented into modifiers (stop-words are discarded). **Step-2**: Adding weighted edges. Given the co-kernel-object elements $CoKO(ko)$, the modifier graph G_{ko} can be obtained by adding an edge for each pair of distinct modifiers mo_i and mo_j iff $\delta_{TKQL}(mo_i, mo_j) > 0$.

Suppose π denotes a parameter-free graph clustering algorithm, given an modifier graph G_{ko}, a group of modifier clusters are generated, say, $MC = \{mc_k\}$, where $mc_k = \{mo_i\}$ denotes a cluster of modifiers. Corresponding to each modifier cluster mc_k, we generate a cluster of subtopic strings, say, $sc_k = \{tStr_m\}$, which is used to represent a subtopic, e.g., $\ddot{t}_k$. Thus, a set of subtopic string clusters $SC = \{sc\}$ can be obtained based on the set of modifier clusters MC. Specifically, for each modifier $mo_i \in mc_k$, we select the subtopic string $tStr_m$ that meets $mo_i \succ tStr_m$, and add $tStr_m$ into the corresponding subtopic string cluster sc_k. For example, given the two modifier clusters by clustering the modifier graph in Fig. 2 (Section 4.1), i.e., $mc_1 = \{$游戏$(game)\}$ and $mc_2 = \{$小说$(fiction)$, 阅读$(reading)\}$, for mc_1, $sc_1 = \{$哈利波特游戏$(HarryPottergame)\}$ will be generated due to 游戏$(game) \succ$ 哈利波特游戏$(HarryPottergame)$, and so does $sc_2 = \{$哈利波特阅读$(HarryPotterreading)$, 哈利波特小说$(HarryPotterfiction)\}$.

When allocating the subtopic string of which the composing modifiers belong to different modifier clusters, we obey the following priority rules: (1) Add this subtopic string into the subtopic string cluster, of which the corresponding modifier cluster encloses more composing modifiers; (2) If two modifier clusters enclose the same number of modifiers of a particular subtopic string, add this subtopic

string into the subtopic string cluster, of which the corresponding modifier cluster encloses the most frequent modifier. The idea of the above rules are straightforward, i.e., a larger number of modifiers means a larger expressive power, and a more frequency modifier carries a larger expressive power than a rare modifier.

4.3 Generating the Ranked List

Definition 3 (Expression Power). *Expression power of a subtopic string is defined as the generating probability of its composing modifiers, which measures its effectiveness of describing the subtopic represented by the subtopic string cluster it belongs to. It is given as:* $EP(tStr) = \prod_{mo_i \succ tStr} p(mo_i)$.

By $p(mo_i) = \frac{|mo_i|+1}{\sum_{\{mo_j \in V\}} |mo_j|+|V|^+}$, where $|mo|$ denotes modifier frequency in V and $|V|^+$ denotes the number of distinct modifiers, we assume that there is a probability distribution for modifiers, which is sampled repeatedly by users to form mutually-independent modifiers to specify kernel-object oriented subtopics.

Definition 4 (Subtopic Popularity). *Subtopic popularity is defined as the likelihood of a particular subtopic. Formally, it is given as* $SP(\ddot{t}_k) = SP(sc_k) = \frac{|sc_k|}{\sum_{sc_s \in SC} |sc_s|}$, *where* $\ddot{t}_k$ *denotes the subtopic represented by the subtopic string cluster* sc_k, $|sc_k| = \sum_{tStr_m \in sc_k} |tStr_m|$ *and* $|tStr_m|$ *denotes the frequency of* $tStr_m$ *in* $CoKO(ko)$.

Based on Definition 4, the gain value of putting a subtopic string $tStr_r$ at the r-th slot is given as: $G(r) = \frac{SP(sc_k)}{log(r+1)}$, where $tStr_r \in sc_k$, i.e., the gain value builds upon the subtopic popularity of the subtopic string cluster that $tStr_k$ belongs to. The discounted cumulative gain of L with a cutoff r is given as: $DCG(r) = \beta * \frac{N_{1+}(r)}{|SC|} + (1-\beta) * \sum_{j=1}^{r} G(j)$, where $N_{1+}(r)$ denotes the number of distinct clusters that the top-r subtopic strings cover. If we view the cluster number $|SC|$ as the number of possible subtopics, $\frac{N_{1+}(r)}{|SC|}$ is used to represent the subtopic diversity. $\sum_{j=1}^{r} G(j)$ aims to rank subtopic strings that indicate popular subtopics at higher positions. Analogous to the metric of *D#-nDCG* [18], we let $\beta = 0.5$. Inspired by the Maximum Marginal Relevance (MMR) [8] criterion that combines relevance and novelty in the context of text retrieval, Algorithm 1 generates the target ranked list L by iteratively selecting the subtopic string that achieves the maximum margin.

In particular, by Steps 1-6, from the subtopic string cluster that achieves the maximum subtopic popularity, the subtopic string that has the maximum expression power is selected as the first element of L. By Steps 7-12, given the top-$(j-1)$ subtopic strings, the subtopic string that achieves the maximum margin (i.e., $DCG(j) - DCG(j-1)$) when being selected as the j-th element of L is put at the j-th slot of L. Because the discounted cumulative gain value (DCG) combines both subtopic diversity and relevance, the obtained L thus captures a trade-off between subtopic diversity and popularity.

Algorithm 1. The algorithm for generating the ranked list L

Input: η: required size of L; $SC = \{sc_k\}$: the set of subtopic string clusters.

1: **for** each $sc_k \in SC$ **do**
2: Rank the subtopic strings of sc_k in order of their decreasing expression power;
3: **end for**
4: Select the subtopic string cluster sc^* that meets: $sc^* = \arg\max_{sc_k \in SC} SP(sc_k)$;
5: Select the subtopic string $tStr^*$ that meets: $tStr^* = \arg\max_{tStr_m \in sc^*} EP(tStr_m)$;
6: Put $tStr*$ at the first slot of the target ranked list L and remove it from sc^*;
7: Set $j = 2$;
8: **while** $j \leq \eta$ *and* $|SC| > 0$ **do**
9: Select $tStr^*$ that meets: $tStr^* = \arg\max_{\{sc_n \in SC\}}(DCG(j) - DCG(j-1))$;
10: Put $tStr^*$ at the j-th slot of L and remove $tStr^*$ from the cluster it belongs to;
11: $j++$;
12: **end while**

5 Experiments

5.1 Experimental Setup

The publicly available topic set and resources (e.g., Chinese query log SogouQ, query suggestions and the ideal ranked list for each topic) from the Intent task of NTCIR-10 [17] are adopted, the top-100 result snippets of Google are used. Yu and Ren [24] have shown that the role-implicit queries are often question queries or verbose queries that merely require a specific answer. In this paper, we focus on role-explicit topics that generally have multiple subtopics, 6 role-implicit topics (i.e., 0244, 0245, 0249, 0256, 0270, 0283) are excluded, e.g., *0256:* 什么是 *RTF*(what is RTF). The remaining 92 topics are used as *Topic-Set-A*.

Fig. 3 (in Section 5.2) shows to what extent we can rely on the QKQL. The x-axes denote the count of instances that appeared in SogouQ. The y-axes denote the count of topics. The asterisk corresponds to co-session queries, the circle corresponds to candidate subtopic strings. We found that: 46 topics have co-session queries less than 10, 38 topics have candidate subtopic strings in SogouQ less than 10. For topics of this kind, the so called *sparseness problem* is a big trouble. By excluding the 38 topics from Topic-Set-A, of which the count of candidate subtopic strings in SogouQ is less than 10, we built the *Topic-Set-B*.

Two typical methods [9, 16] for query intent inference are compared. Deng et al. [9] proposed the *Click Frequency Inverse Query Frequency model* for query clustering (denoted as *CFIQF*). Under CFIQF, each query is represented by a vector of clicked documents. Sadikov et al. [16] performed intent inference by clustering the refinements of a given query (denoted as *RC*). A Markov graph is built based on the document click and session co-occurrence of the refinements. Finally, the problem of refinement clustering is reduced to the problem of Euclidean-vector clustering. As did by Sadikov et al. [16], the complete-linkage clustering algorithm is used for the two baselines.

The metric *D#-nDCG* [18] suggested by NTCIR-10 [17] is used as the test metric, which is a linear combination of *I-rec* and *D-nDCG*. I-rec indicates the

proportion of subtopics covered by the top subtopic strings and measures diversity. D-nDCG measures overall relevance across subtopics. In our study, we evaluate I-rec, D-nDCG, D#-nDCG with cutoff values of 10 and 20.

When quantifying the edges of a modifier graph, we let $\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{3}$, i.e. co-query, term-level co-session and term-level co-click were treated equally. For modifier graph clustering, two parameter-free graph clustering algorithms *Louvain* [3] and *LinLog* [12] have been tested, the corresponding modifier graph based approaches are denoted as *MG-Lou* and *MG-Lin* respectively.

5.2 Experimental Results

Based on Topic-Set-B, Table 3 shows the results with cutoff values of $l = 10\&20$.

Table 3. Topic-Set-B based comparison with cutoff values of $l = 10\&20$

	I-rec@10	D-nDCG@10	D#-nDCG@10	I-rec@20	D-nDCG@20	D#-nDCG@20
CFIQF(5)	0.1782	0.1753	0.1768	0.1782	0.1168	0.1475
CFIQF(10)	0.3066	0.2688	0.2877	0.3066	0.1767	0.2417
CFIQF(15)	0.3093	0.2715	0.2904	0.3903	0.2328	0.3116
RC(5)	0.0353	0.0333	0.0343	0.0353	0.0216	0.0285
RC(10)	0.0595	0.0501	0.0548	0.0595	0.0327	0.0461
RC(15)	0.0806	0.0672	0.0739	0.1024	0.0504	0.0764
MG-Lou	0.4375	**0.4375**	0.4375	0.5579	**0.3975**	0.4777
MG-Lin	**0.4650**	0.4275	**0.4462**	**0.5878**	0.3921	**0.4899**

As the complete-linkage clustering algorithm requires a predefined cluster number, different values (e.g., 5 is marked as (5)) were tested for the baselines. As shown in Table 3, the baseline methods exhibit different performances due to different cluster numbers. For example, the D#-nDCG@10 value of CFIQF is 0.2904 (4th column) with a cluster number of 15, which is better than the values generated with cluster numbers of 5 and 10. However, the number of subtopics indeed varies from topic to topic. A predefined cluster number can't guarantee a natural subtopic mining. RC operates by clustering the query refinements (essentially co-session queries) of a given topic, its low performance implies that the clustered refinements greatly deviates from the ranked list by human assessors. By clustering the subtopic strings via co-click information, CFIQF shows a better performance than RC.

Different from the baseline methods that rely on QKQL, the modifier graph based approaches make use of TKQL. The results show that both MG-Lou and MG-Lin outperform the baselines in terms of I-rec, D-nDCG and D#-nDCG. The reasons are: The proposed approaches work at a term level instead of a whole query level. Based on the TKQL, they can determine the subtopics of subtopic strings from different resources rather than merely the queries of a query log. For example, though 哈利波特阅读(Harry Potter reading) (Table 1) is not included in SogouQ, we can determine its underlying subtopic based on its composing modifier 阅读(reading). On the contrary, the baseline methods will fail due to no click information. Leveraging on the two ad-hoc graph clustering algorithms,

our modifier graph based approaches can determine the possible subtopics per topic instead of a uniform cluster number.

Based on Topic-Set-A, Table 4 shows the results with cutoffs of $l = 10\&20$.

Table 4. Topic-Set-A based comparison with cutoff values of $l = 10\&20$

	I-rec@10	D-nDCG@10	D#-nDCG@10	I-rec@20	D-nDCG@20	D#-nDCG@20
CFIQF(5)	0.1130	0.1042	0.1086	0.1130	0.0714	0.0922
CFIQF(10)	0.1449	0.1352	0.1401	0.1449	0.0890	0.1169
CFIQF(15)	0.1506	0.1430	0.1468	0.2142	0.1242	0.1692
RC(5)	0.0145	0.0160	0.0153	0.0145	0.0104	0.0125
RC(10)	0.0391	0.0355	0.0373	0.0391	0.0230	0.0311
RC(15)	0.0411	0.0373	0.0392	0.0543	0.0282	0.0413
MG-Lou	0.4111	**0.4253**	**0.4182**	**0.5334**	**0.4024**	**0.4679**
MG-Lin	**0.4121**	0.4117	0.4119	0.5309	0.3945	0.4627

Compared with Topic-Set-B, Topic-Set-A contains 38 topics that have sparse QKQL. This is the very reason why the performances of the baseline methods are greatly impacted. For example, the best performance of CFIQF (D#-nDCG@20 at the 7th column) is decreased from 0.3116 (Table 3) to 0.1692 (Table 4). In contrast, our modifier graph based approaches exhibit stable performances. For example, the best performances (results in bold) in Tables 3 and 4 do not differ a lot, which demonstrates that the modifier graph based approaches are robust to the sparseness problem.

To understand the impact that different graph clustering algorithms may have on modifier graph clustering, Fig. 4 illustrates the cluster number (i.e., the number of subtopic string clusters) distribution based on Topic-Set-A. The asterisk represents the official number of subtopics per topic. The circle and triangle represent the cluster number per topic under the Louvain algorithm and the LinLog algorithm respectively.

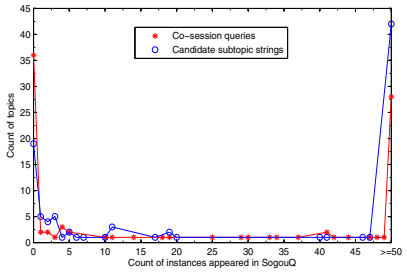

Fig. 3. The extent we can rely on QKQL

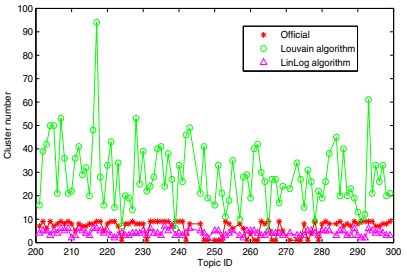

Fig. 4. Cluster number distribution

From Fig. 4, we found that: The two algorithms group the same modifier graph into different number of clusters due to different optimization criteria, and the LinLog algorithm generally outputs smaller numbers of clusters. Unfortunately, both of the two algorithms commonly generate different cluster numbers compared with the official subtopic number. If we directly regard the cluster number

as the subtopic number (as we did in this paper), the approximated subtopic recall value would not be accurate enough. A particular algorithm should be devised for better clustering the modifier graph.

6 Conclusions and Future Work

In this paper, we performed subtopic mining via modifier graph clustering, which makes use of TKQL rather than QKQL. Compared with the baselines that treat a whole subtopic string or query as the minimum analysis unit, our modifier graph based approaches achieve a better performance in terms of I-rec, D-nDCG and D#-nDCG. A limitation of our current study is that the proposed approach is tested against Chinese topics. Whether it works effectively against English topics is planned as the future work. Moreover, devising a specific algorithm for modifier graph clustering would be an interesting future research direction.

Acknowledgements. This research has been partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (A), 22240021.

References

[1] Agrawal, R., Gollapudi, S., Halverson, A., Ieong, S.: Diversifying search results. In: Proceedings of the 2nd WSDM, pp. 5–14 (2009)
[2] Beeferman, D., Berger, A.: Agglomerative clustering of a search engine query log. In: Proceedings of the 6th KDD, pp. 407–416 (2000)
[3] Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics (2008)
[4] Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., Vigna, S.: The query-flow graph: model and applications. In: Proceedings of the 17th CIKM, pp. 609–618 (2008)
[5] Bonchi, F., Perego, R., Silvestri, F., Vahabi, H., Venturini, R.: Recommendations for the long tail by term-query graph. In: Proceedings of the 20th WWW, pp. 15–16 (2011)
[6] Bonchi, F., Perego, R., Silvestri, F., Vahabi, H., Venturini, R.: Efficient query recommendations in the long tail via center-piece subgraphs. In: Proceedings of the 35th SIGIR, pp. 345–354 (2012)
[7] Cao, B., Sun, J.T., Xiang, E.W., Hu, D.H., Yang, Q., Chen, Z.: PQC: personalized query classification. In: Proceedings of the 18th CIKM, pp. 1217–1226 (2009)
[8] Carbonell, J., Goldstein, J.: The use of mmr, diversity-based reranking for reordering documents and producing summaries. In: Proceedings of the 21st SIGIR, pp. 335–336 (1998)
[9] Deng, H., King, I., Lyu, M.R.: Entropy-biased models for query representation on the click graph. In: Proceedings of the 32nd SIGIR, pp. 339–346 (2009)
[10] Hu, Y., Qian, Y., Li, H., Jiang, D., Pei, J., Zheng, Q.: Mining query subtopics from search log data. In: Proceedings of the 35th SIGIR, pp. 305–314 (2012)

[11] Jones, R., Klinkner, K.L.: Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In: Proceedings of the 17th CIKM, pp. 699–708 (2008)
[12] Noack, A.: Energy models for graph clustering. Journal of Graph Algorithms and Applications 11(2), 453–480 (2007)
[13] Radlinski, F., Dumais, S.: Improving personalized web search using result diversification. In: Proceedings of the 29th SIGIR, pp. 691–692 (2006)
[14] Radlinski, F., Szummer, M., Craswell, N.: Inferring query intent from reformulations and clicks. In: Proceedings of the 19th WWW, pp. 1171–1172 (2010)
[15] Ren, F., Sohrab, M.G.: Class-indexing-based term weighting for automatic text classification. Information Sciences 236, 109–125 (2013)
[16] Sadikov, E., Madhavan, J., Wang, L., Halevy, A.: Clustering query refinements by user intent. In: Proceedings of the 19th WWW, pp. 841–850 (2010)
[17] Sakai, T., Dou, Z., Yamamoto, T., Liu, Y., Zhang, M., Song, R.: Overview of the NTCIR-10 INTENT-2 task. In: Proceedings of NTCIR-10 Workshop, pp. 94–123 (2013)
[18] Sakai, T., Song, R.: Evaluating diversified search results using per-intent graded relevance. In: Proceedings of the 34th SIGIR, pp. 1043–1052 (2011)
[19] Song, R., Zhang, M., Sakai, T., Kato, M.P., Liu, Y., Sugimoto, M., Wang, Q., Orii, N.: Overview of the NTCIR-9 INTENT task. In: Proceedings of NTCIR-9 Workshop Meeting, pp. 82–105 (2011)
[20] Song, Y., Zhou, D., He, L.: Query suggestion by constructing term-transition graphs. In: Proceedings of the 5th WSDM, pp. 353–362 (2012)
[21] Wang, X., Chakrabarti, D., Punera, K.: Mining broad latent query aspects from search sessions. In: Proceedings of the 15th KDD, pp. 867–876 (2009)
[22] Wen, J.R., Nie, J.Y., Zhang, H.J.: Clustering user queries of a search engine. In: Proceedings of the 10th WWW, pp. 162–168 (2001)
[23] Yin, X., Shah, S.: Building taxonomy of web search intents for name entity queries. In: Proceedings of the 19th WWW, pp. 1001–1010 (2010)
[24] Yu, H., Ren, F.: Role-explicit query identification and intent role annotation. In: Proceedings of the 21st CIKM, pp. 1163–1172 (2012)