Categorizing Search Results Using WordNet and Wikipedia

Reza Taghizadeh Hemayati[1], Weiyi Meng[1], Clement Yu[2]

[1] Department of Computer Science, Binghamton University,
Binghamton, NY 13902, USA
{hemayati, meng}@cs.binghamton.edu
[2] Department of Computer Science University of Illinois at Chicago,
Chicago, IL 60607, USA
cyu@uic.edu

Abstract. Terms used in search queries often have multiple meanings and usages. Consequently, search results corresponding to different meanings or usages may be retrieved, making identifying relevant results inconvenient and time-consuming. In this paper, we study the problem of grouping the search results based on the different meanings and usages of a query. We build on a previous work that identifies and ranks possible categories of any user query based on the meanings and common usages of the terms and phrases within the query. We use these categories to group search results. In this paper, we study different methods, including several new methods, to assign search result record (SRRs) to the categories. Our SRR grouping framework supports a combination of categorization, clustering and query rewriting techniques. Our experimental results show that some of our grouping methods can achieve high accuracy.

Keywords: Search Result Clustering and Categorization, Search Engine

1 Introduction

One common complaint about current search engines is that they return too many irrelevant results for users' queries. The reasons include (1) current search engines retrieve results mainly based on query words match, capturing only the main meanings or usages of query words, and (2) Internet users tend to submit very short queries which often do not provide enough context to determine the users' intentions. One way to tackle this problem is to group the search results into multiple categories such that all results in the same category correspond to the same meaning or usage of the query. This makes it much easier for users to identify useful results. Most current result clustering techniques are based on word-match similarity. Although a few techniques have used semantic similarity [1, 2], they have various weaknesses. For example, they do not explicitly and systematically consider usages of query terms, which would lower the quality of search result clustering.

In this paper, we propose and evaluate several methods to assign search results to a type of categories called *definition categories* (DCs). DCs are obtained based on both the possible meanings and the usages of the terms and/or phrases in a query using the techniques proposed in our previous paper [5].

Unlike the work in [4] which considered only single-term queries, in this paper we consider both single-term and multi-term queries. Furthermore, the categories used in

this paper are defined based on both the possible meanings and usages of the terms and/or phrases in the query using WordNet and Wikipedia [5]. The categories are ranked based on both the importance of the meanings/usages of each term/phrase in the query and the relationships between them. Moreover, we also introduce a new method which uses query rewriting technique to categorize SRRs.

This paper has the following contributions:

1. We introduce three new automatic real-time grouping methods. The first one is based on a query-rewriting technique (QRW). This method selects some query expansion terms (QET) and submits these QETs along with the original query to search engine(s) to retrieve related results for each category (DC). The second method (E3C) is extension to the CCC algorithm first introduced in [4]. E3C is designed to improve assigning SRRs that have low similarities with a DC to the right DC. The third method is a hybrid of the first two methods. It decides which of the first two methods to use in different situations in order to achieve better overall performance.
2. We perform extensive experiments to evaluate and compare the performance of our proposed algorithms. The experimental results indicate that some of the proposed algorithms have both good effectiveness and efficiency.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 provides an overview of our approach. Section 4 presents the main steps of our approach. Section 5 reports experimental results. Section 6 concludes the paper.

2 Related Work

The general problem of document clustering and categorization has been studied extensively [8] and they will not be reviewed in this paper. Instead, we focus on related works that deal with the clustering and categorization of the search result records (SRRs) returned from search or metasearch engines.

Techniques for clustering web documents and SRRs have been reported in many papers and systems such as [2, 15, 17, 18]. There are also commercial search engines like yippy.com which clusters SRRS. However, these techniques perform clustering based on the syntactic similarity but not semantic similarity.

Some researchers used web directories like Yahoo directory or ODP to categorize/classify user queries. Mapping user queries to hierarchical sequences of topic categories was studied in [3]. The method in [9] maps user queries to categories using a user profile learned from the user's search history and a general profile derived from a concept hierarchy. The method in [16] classifies the search results into deep hierarchies using category candidates retrieved by query.

Another related area is *result diversification* (e.g., [13, 1]), which aims to select search results covering different meanings/usages and show them among the top-ranked results. These methods do not specifically cluster or categorize search results.

Query disambiguation [10] is relevant to identifying different meanings of a query for generating different categories. The issue of generating categories is not considered in this paper as our methods are based on already available categories.

Techniques for clustering and categorizing web documents using WordNet or other ontologies have also been extensively studied (e.g., [11, 14, 7, 15]) and some of them (e.g., [11, 14]) also tried to categorize SRRs based on the meanings of the query term.

Our approach differs from the above techniques significantly. First, our SRR grouping algorithm employs categorization, clustering and query rewriting techniques in a unique way. Second, our method also copes with SRRs that do not match any meaning/usage of the query term/phrase in WordNet or Wikipedia definitions. In other words, we utilize definitions provided by WordNet and Wikipedia but are not limited by them. Third, our approach also utilizes similarities that are computed using syntactical, semantic and common usage information.

3 System Overview

Our overall search result grouping system consists of the following main components:

1. *Alternative query generation*. For each user query *Q*, this step generates a set of *alternative queries* (AQs). All AQs contain the same set of query terms that appear in *Q* but contain different phrases. This step identifies different possible phrases comprised of the terms in *Q* as phrases are better at capturing the meanings of user queries. We will refer both query terms and phrases as ***concepts***.

2. *Definition category generation*. A *definition category* (DC) is a combination of *meanings* or *usages* derived from the concepts of an AQ. This step generates all possible DCs for each AQ and it consists of three tasks: (a) *Usage generation*: Identify all possible meanings/usages for each concept in AQ using semantic dictionaries (WordNet and Wikipedia). (b) *Usage merging*: Merge similar meanings or usages for each concept into a single meaning/usage to reduce possible confusion to users. (c) *Definition category generation*: Generate DCs by combining one (possibly merged) meanings/usage from each concept in the AQ.

3. *Definition category ranking*. This step ranks the generated DCs. Each *DC* is generated from a specific alternative query *AQ**, from a specific meaning/usage of each concept in *AQ**, and from the combination of these specific meanings/usages. To rank the DCs, each DC is weighted in three aspects: (a) the importance of its AQ among all Aqs; (b) the importance of each DC among all DCs within each AQ; and (c) the document frequncy of each DC on the Web.

4. *Submitting query and processing results:* For each user query, the top k (k = 50 in this paper) distinct results (duplicates are removed) are retrieved and are used as input to our SRR grouping algorithm (next component). Each result (SRR) usually consists of three different items: title, URL and snippet. Only the title and snippet of each SRR will be utilized to perform the grouping in our current approach. For each SRR, we first remove the stop words and stem each remaining word. Next, the SRR is converted as a vector of terms.

The above components were introduced in our previous works [4, 5] and will not be repeated in this paper. This paper focuses on the result grouping component as briefly reviewed below.

5. *SRR grouping algorithms:* We evaluate four major grouping algorithms in this paper (CCC, QRW, E3C and Hybrid). Our CCC and E3C algorithms consist of the following three steps to group SRRs: (i) Preliminary **C**ategorization, (ii) Further Categorization, and (iii) Final Categorization. We explain these in more detail in section 4. The CCC algorithm here is similar to the CCC algorithm introduced in our previous work [4]. In [4], only WordNet was utilized to categorize SRRs and only

single-term queries were considered. In this paper, we use both WordNet and Wikipedia to categorize SRRs and both single-term and multi-term queries are considered. The main difference between CCC and E3C is the similarity computation method used in the second step. The new method is not only based on the similarity between SRRs and DCs (categories) but also the similarities between un-categorized SRRs and already categorized SRRs. We will explain this in more details in Section 4.

Query ReWriting (QRW) uses the query expansion terms QETs generated from DCs along with the original query as new queries to retrieve SRRs related to each DC. We will discuss this in more details in 4.3. Hybrid solution uses one of the QRW or CCC [4] approaches based on the type of the DCs generated from the submitted query. Our experiments (section 5) show that this approach improves performance.

4 SRR Grouping Algorithms

We present four algorithms to group SRRs (CCC, E3C, QRW, Hybrid) in this section.

4.1 Algorithm CCC

Algorithm CCC in this paper consists of three major steps:
Step1: Categorize SRRs by assigning each SRR to the most similar DC if the similarity is greater than a threshold T_1. Temporary categories are obtained based on the current assignments and the remaining SRRs form another temporary category.
Step2: Further categorize the remaining SRRs by assigning each such SRR to the most similar temporary category using another threshold T_2.
Step3: Categorize/Cluster the set *RS* of remaining SRRs. Three alternative solutions were introduced for this step in [4]: (i) *Largest Frequency of Use* (*LF*): Assign RS to the cluster with the most common meaning; (ii) *Largest Category*: Assign RS to the cluster (from Step 2) with the largest size; (iii) *Clustering*: cluster the results in RS.

The CCC algorithm in this paper differs from that in [4] in the following aspects:

1. DCs used in this paper are based on both WordNet and Wikipedia definitions; however the one in [4] was just based on only WordNet definitions.
2. In this paper, we train thresholds in different steps of CCC to achieve the best performance. In [4] the thresholds were manually selected.
3. In this paper, the *LF* method in [4] is replaced by an HWC method, which assigns RS to the DC with the highest weight. *LF* is based on the largest *frequency of use* of the query term's synset in WordNet. HWC is based on the highest weighted (ranked) DC (definition category) obtained using both WordNet and Wikipedia (see [5] for details).

4.2 E3C (Extended CCC)

Algorithm **E3C** has the same three major steps as CCC except that the second step (i.e., Further Categorization) in E3C is implemented differently. In CCC, sometimes we couldn't assign a related SRR to the correct DC since we couldn't find high similarity between that SRR and the DC due to the lack of sufficient common words. To address this issue, we modify the second step in CCC. This is explained below.

Let's assume that there are similarity scores among documents and also between each document in a set of documents (SRRs) and a DC C_1. It is possible that an SRR R_1 has zero or very low similarity with C_1, but is still sufficiently similar to an SRR R_2

which is very similar to C_1 and has already been assigned to C_1. In this case we may assign R_1 to C_1 via R_2.

We categorize SRRs based on their $V(R, C_k)$ values (to be defined shortly) with DCs. Specifically, for each SRR R, find the DC C_k that has the highest $V(R, C_k)$ value to R among all DCs. If the value of $V(R, C_k)$ is very low, we postpone assigning R to a later step. This is to prevent assigning an SRR to a DC with a very low similarity. When more information becomes available later, we will try to assign this SRR again. If the $V(R, C_k)$ values between R and two DCs are very similar, it is easy to assign R to a wrong cluster. In this case, we also postpone assigning R to a later step.

We continue this method until one of the following conditions becomes true:

- All SRRs have been categorized.
- We cannot assign at least one new SRR to any DC.
- A pre-set number of iterations have been reached. In this case we don't continue assigning un-assigned SRRs to DCs and go to Step 3 of CCC.

We introduce two approaches to calculate $V(R, C_k)$ values. In the first approach, the value of $V(R, C_k)$ is calculated by

$$V(R, C_k) = Sim(R, C_k) + \sum_{i=1}^{n} Sim(R, d_i) P(d_i \mid C_k)$$

where $P(d_i \mid C_k) = \frac{Sim(d_i, C_k)}{\sum_{j=1}^{m} Sim(d_i, C_j)}$ and $1 \leq k \leq m$ and $Sim(R, C_k)$ is the similarity between R and the kth DC, $Sim(R, d_i)$ is the similarity between R and d_i which is the ith already categorized SRR in DC C_k, m is the number of DCs built for a submitted query, $P(d_i|C_k)$ is the probability of d_i belonging to C_k (the kth DC), $Sim(d_i, C_k)$ is the similarity between d_i, which is the ith already categorized SRR in DC C_k, and C_k.

We consider these values in this method. First, the similarity between R and C_k. Second, the total similarities between R and the other SRRs in C_k ($1 \leq k \leq m$). This will be determined by calculating the similarity between R and each SRR d_i in C_k multiplied by the probability of d_i belonging to C_k. By using this probability, we are more in favor of those assigned SRRs that have higher similarities to a DC over those with lower similarities to the same DC.

The reason behind this approach is, if there are similarities between an SRR R and SRRs already categorized in a DC C_k and the total value of these similarities and the similarity between R and C_k is above a threshold (e) and is the highest among all DCs, then there is a good chance that R belongs to C_k.

The second approach determines the value of $V(R, C_k)$ by using:

$$V(R, C_k) = Max\left(Max_{i=1}^{n}(Sim(R, d_i) * P(d_i \mid C_k)), \alpha * Sim(R, C_k)\right)$$

where $0 \leq \alpha \leq 1$. In this method we determine $V(R, C_k)$ by finding the maximum value among the similarity between R and SRR d_i in C_k multiplied by the probability of d_i belonging to C_k and the similarity between R and C_k. The logic behind this method is if an SRR R_1 has high similarity with an SRR R_2 which has already been categorized to a DC C_1, and also this R_2 has high similarity to DC C_1, then there is a good chance that this R_1 (uncategorized yet) belongs to C_1, compared to a situation that the same SRR R_1 has similarity to more SRRs (e.g., R_3, R_4) in another DC C_2, but the similarity between R_1 and R_3 (or R_4) multiplied by the probability of their (R_3 or R_4) belonging to

C_2 is lower compared to the first case when R_1 is similar to R_2 and R_2 has been categorized into C_1.

4.3 Query ReWriting (QRW)

The last two algorithms send original queries and retrieve the first *n* SRRs (first 50 SRRs in this work). In those algorithms, usually high weighted DCs get relevant SRRs and DCs with lower weights don't receive any SRRs. In order to get relevant SRRs for low-weighted DCs, we need to retrieve many SRRs (in many cases thousands). To address this issue, we introduce the QRW method.

In this method, we first find a set of query expansion terms (QETs) for each DC (category), and send them along with the original query to a search engine to retrieve relevant SRRs for each DC. This will guarantee that there won't be any DCs without SRRs. Since our DCs are built based on Wikipedia and WordNet meanings/usages definitions, there is a very low chance to not retrieve any relevant SRRs for any DC. This method has the following steps: Send the original query to the DC generator, retrieve DCs, find query expansion terms (QET) for each DC, and finally send each DC's QETs along with the original query as a new query to a search engine to retrieve relevant SRRs for each DC. We now explain how to find the QETs for DCs (the first two steps have been discussed in Section 3 and our previous paper [5]). In order to find the best expansion terms for DCs we use two sources: Wikipedia and WordNet.

Due to space limitation, we cannot provide the details of our QET generation algorithm in this paper. The basic idea is sketched below. Given a DC as input, this method aims to obtain query expansion terms to represent the DC. QETs for a DC are a set of terms/phrases that summarizes the DC. We first use Wikipedia and WordNet to generate candidate QETs. Wikipedia provides some useful information for each concept (like meanings/usages definitions, categories, disambiguation page and etc.), which can be used to generate candidate QETs. WordNet provides information like synonyms, hypernyms, and etc. that can also be used as candidate QETs. These candidate QETs will be ranked for each DC based on their similarities and the top-ranked candidate QETs is then chosen as the final QETs for the DC.

4.4 Hybrid Method

We observed that algorithm E3C outperforms the QRW method for certain types of DCs while the opposite is true for other types of DC's (more details will be given shortly). We introduce a hybrid solution to take advantage of the strengths of both types of approaches to enhance the performance of search result grouping. The following are the main differences between E3C and QRW.

1. QRW focuses only on information retrieved from Wikipedia and WordNet, while E3C uses information retrieved from WordNet, Wikipedia and SRRs.
2. For DCs that do not contain a single phrase or for DCs with weak relationships between their concepts and meanings/usages, it is often difficult to determine a single meaning/usage which can express the intention of the DC. As a result, it is more difficult to find good QETs for these DCs. For this type of queries, E3C is more applicable than QRW.
3. Using Wikipedia and WordNet to categorize results places more emphasis on what meanings/usages are covered by a dictionary/encyclopedia for a concept. On the

other hand, using SRRs to categorize the results emphasizes more on what contents are indexed by a search engine.

4. The importance of different meaning/usage recognized by a dictionary/ encyclopedia and SRRs can be different. A good balance between them can help the system categorize and rank SRRs with better accuracy and users' satisfaction. We try to achieve this goal by exploring different grouping algorithms.
5. Although Wikipedia is a dynamic source, SRRs are more dynamic and up-to-date. Just using Wikipedia to find possible meanings/usages may miss some usages (like persons' names). Using all sources (Wikipedia, WordNet and SRRs) makes our system capable of addressing this concern.
6. In CCC/E3C, depending on the number of SRRs retrieved, we may have some DCs with no SRRs assigned. In order to have at least some SRRs in each DC, we may need to retrieve thousands of SRRs. QRW does not suffer from this problem.

We differentiate four types of DCs. Each type of DC determines how well we can "guess" the real intention of a user by determining a meaning/usage for a user query. Based on our experiments, the QRW algorithm performs better for DCs for which we can find a meaning/usage, which can express the whole DC compared to other algorithms (e.g., when there is a definition page for the DC in Wikipedia which can be used as the DC's representative). On the other hand, E3C performs better for DCs for which we cannot find a single meaning/usage to represent those DCs (e.g., there is no definition page in Wikipedia which can be used as the DC's representative when sending the corresponding AQ to Wikipedia). Based on which algorithm (E3C or QRW) we use, different SRRs will be categorized and they will be categorized differently. For E3C, SRRs from the original query will be assigned to the similar DCs, but for QRW, SRRs retrieved by queries formed by QETs along with the original query for each DC will be assigned to the corresponding DC.

In the hybrid algorithm, we use a method to select one of the algorithms (E3C or QRW) based on the type of DC generated from a query submitted by a user. We define four types of DCs. We also define different cases for each type of DC. These cases are used to determine the type of a DC by examining different definitions (if exist) in each DC to see if there is a single meaning/usage which can be used on behalf of other meanings/usages in a DC. Each query may consist of different concepts and each concept may have different meanings/usages. Therefore each DC may contain different meanings/usages. We try to examine each of these meanings/usages to see if there is relationship between them so we can consider all meanings/usages in a DC that are related to each other.

We classify DCs into the following four types:

Type 1*: Single term Queries with single meaning.*

Case 1: A DC generated from an AQ (alternative query explained in section 3 step 1) which is a single term query with a single meaning/usage.

Type 2*: DCs with no ambiguity.* This type of DCs has only one meaning/usage (the DC contains only one meaning/usage) from both WordNet and Wikipedia after merging similar meanings/usages [5].

We recognize a DC as *Type 2* if one of the following cases is true:

Case 1: A DC generated from an AQ which is a single term query with multiple meanings/usages from WordNet/Wikipedia. This means we can see multiple

definition pages (there are multiple meanings/usages for this concept) when we submit this query to Wikipedia/WordNet. This type of AQs will generate multiple DCs, but each DC refers to one meaning/usage directly from WordNet/Wikipedia. All DCs generated from this AQ are *Type 2* in this case.

Case 2: A DC generated from an AQ which is a valid phrase [5] with a single or multiple meanings/usages from Wikipedia/WordNet. Each DC refers to one meaning/usage directly from WordNet/Wikipedia. All DCs generated from this AQ are *Type 2* in this case.

Type 3*: DCs with multiple definitions, but there is one definition which can be used to represent all other definitions in a DC*. This type of DCs has multiple meanings/usages in WordNet or Wikipedia; however we can find one meaning /usage that can be used to represent all other meanings/usages in a DC. For example, if all meanings of a DC are synonyms, then one can be used to represent the others. As another example, if one meaning is a hypernym of other meanings, then this meaning can represent the others. We have identified 12 cases in which one meaning/usage can represent all other meanings/usages. But due to space limitation, they are not included here. These cases plus many examples for these cases can be found in a technical report [6].

Type 4*: DCs with ambiguity: There is no single meaning/usage which can be used to represent all other meanings/usages in a DC*. We recognize a DC as *Type 4* if it is not recognized one of the above three types.

The QRW method tends to generate longer queries for DCs in *Type 4* compared to DCs of *Types* 1, 2 and 3. The reason is that the system couldn't recognize one single meaning/usage for this type of DC. This type of DC has multiple meanings/usages. Each meaning/usage in a DC generates a set of expansion terms (explained in section 4.3). This usually will result in long and inaccurate QETs (query expansion terms). Furthermore, these QETs may retrieve irrelevant SRRs. Therefore, the Hybrid method uses the E3C algorithm for *Type 4* DCs and uses QRW algorithm for DCs of *Types 1, 2 and 3*.

5 Evaluation

5.1 Query set and performance measures

Our query set contains 50 queries [6] with 25 from TREC (2003 and 2005) and 25 from AOL query logs. We sample 50 queries conditioned on: (a) the query set should have queries with different lengths, (b) the query set should have a mixture of queries with/without phrases, and (c) the query set should have queries with ambiguities. The reason for using queries from different sources is to have queries with different lengths and also have enough queries with ambiguities.

We evaluate four SRR grouping algorithms (CCC, E3C, QRW and Hybrid). For all algorithms, we use the *recall*, *precision* and *F1 measure* as the performance measures. For the SRR grouping algorithms, the recall and precision are defined in [4].

5.2 Performance of CCC and E3C in Different Steps

The CCC algorithm has three major steps to assign (categorize) SRRs to DCs. We evaluate the performance of each step. For Step 3, different methods are introduced to

categorize unassigned SRRs. We also study the performance of each of these methods here. We evaluate the following four different methods for Step 3: Clustering, Largest Cluster (LC), Highest Weighted Cluster (HWC) and SIM (classification).

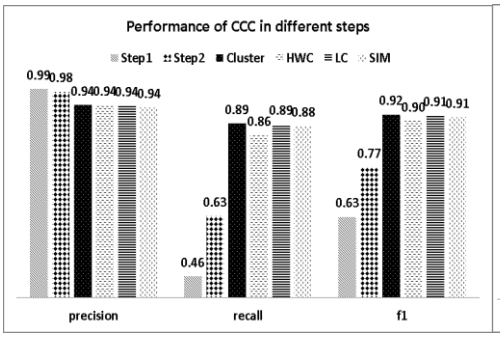

Fig. 1.Performance of CCC

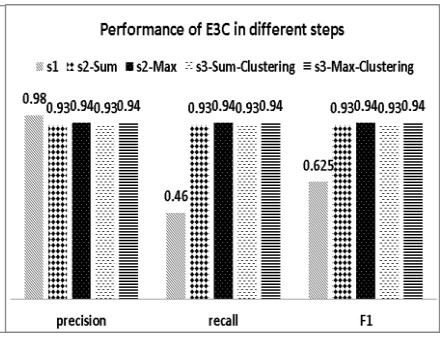

Fig. 2. Performance of E3C

Fig. 1 shows the performance of CCC in different steps. We can observe that the precision of step 1 (p=0.99) is higher than the later steps. The reason is that we try to only assign those SRRs that have very high similarities to DCs. This will decrease the chance of assigning SRRs to wrong DCs. Among different methods for Step 3, clustering method performs the best considering both precision (0.94) and recall (0.89). This is mainly due to the fact that clustering method can group the SRRs beyond the discovered DCs while the other methods force the SRRs that do not match any DCs into incorrect categories. On the other hand, the recalls of earlier steps are very low compared to the later ones. The recalls for step 1, step 2 and step 3 are 0.46, 0.63 and > 0.85 (depending on what method was chosen for step 3). This is due to the fact that in the last step we try to categorize all SRRs and this will increase the recall value compared to earlier steps.

The E3C algorithm has three major steps to assign SRRs to the DCs built for a submitted query. We evaluate the performance of each step of the E3C algorithm here (Fig. 2). Step 1 in E3C is similar to that in CCC. In step 2 of E3C, we introduced two different approaches (Section 4.2). From Fig. 2, we can observe that the precision of step 1 (0.98) is higher than the later step. The reason is the same as that for algorithm CCC. In step 2, the second approach (i.e., when maximum value of $V(R,C_k)$ was chosen to assign an SRR to a DC) performs slightly better than the first approach (p=0.94 vs. p=0.93). We observe that for most cases both approaches categorize SRRs the same way. For cases that there were differences, the second approach tends to perform better. The reason is that if an SRR is very similar to an already assigned SRR to a DC, there is good chance that this SRR belongs to this DC, compared to a situation that the same SRR has moderate similarity with more SRRs in another DC, but the similarity with each of them is much lower compared to the one in another DC. On the other hand, the recall of earlier step (step 1) is very low compared to the later steps. The recalls for step 1, step 2-first approach (Sum) and step 2-second approach (Max) are 0.46, 0.93 and 0.94, respectively. This is due to the fact that in the second step we are less conservative and we give more chances to categorize SRRs to similar DCs since we have more information for each DC in this step (i.e., the SRRs assigned in Step 1 can be utilized). The performance in the third step stays almost the

same as the performance in the second step since we were able to categorize most of the SRRs by the end of step 2 in E3C.

When comparing the performance of the second step in CCC and E3C, we can observe that the precision of CCC in step 2 (0.98) is better than the ones in E3C. On the other hand, the recall of the second step of CCC (0.63) is the lowest compared to the ones in E3C. In E3C, we assign more SRRs to DCs since we don't just assign SRRs to DCs based on their similarities to DCs but also based on the similarities with already assigned SRRs. The precision and recall of E3C in the second step for the first and the second approaches are 0.93 and 0.94. The precision of CCC in the second step is better than the ones in E3C because we only assign very similar SRRs to DCs. When CCC were used there were still uncategorized SRRs at the end of step 2. When E3C was used, we were able to categorize most of the SRRs at the end of step 2. In E3C there were few cases that we assigned SRRs to wrong DCs due to the fact that they either belong to DCs that are not generated by the DC generation algorithm in [5] or we assign them to the wrong DCs due to the lack of common words. Furthermore, in the second step of CCC we can have different thresholds. In the second step of CCC, we can be conservative and keep the value close to the value in the first step ($T_1 \geq T_2$) by decreasing the threshold value very little. In this case we only assign very similar SRRs to DCs. On the other hand, we can be more liberal by decreasing the threshold value more significantly from its original value in the first step in order to assign more SRRs to DCs. We observe that CCC performs better when more conservative approach is chosen. For example the precision, recall and F1 of CCC when clustering is chosen for more conservative approach are 0.94, 0.89 and 0.91, respectively, while for more liberal approach (the threshold value is much lower comparing to the conservative approach ($T_1 >> T_2 > 0$)), these values are 0.94, 0.79 and 0.86, respectively. We consider an approach to be more conservative when T_2 is closer to T_1, and more liberal when T_2 is closer to zero.

5.3 Overall Performance of All Algorithms

We introduced four algorithms to group SRRs in this paper (CCC, E3C, QRW and Hybrid). We study the performance of each of these algorithms here. Figure 3 shows the overall performance of different algorithms discussed in this paper.

To study the performance of CCC in this section, we consider the performance of the algorithm when clustering method is chosen in step 3, since it is the best among all the options. Similarly in E3C we consider the Max method in the second step since it performs better than the Sum method.

QRW performs better than CCC (F1 in QRW is 0.94 vs. F1 in CCC is 0.91). Hybrid method, which is combination of E3C and QRW algorithms, performs the best among different algorithms (precision is 0.96 and recall is 0.94). This is due to the fact that the system predicts what method performs better based on the submitted query. On the other hand, the Hybrid method needs more time to determine the types of the queries (its complexity is linear).

E3C performs better than CCC especially when we consider the recall (the recall for E3C is 0.94 vs. 0.89 for CCC, the precision for both algorithms are 0.94). This is more due to the fact that we were able to categorize more SRRs correctly for higher weighted categories. Based on our observation, E3C also is a faster algorithm (fewer

iterations), although its complexity stays the same. Hybrid still performs slightly better than E3C (F1 is 0.95 for Hybrid vs. 0.94 for E3C).

The results show that all the four algorithms studied in this paper achieved significantly higher accuracy than a *Pure Clustering* method (this method just clusters all SRRs without considering DCs. The number of clusters is set to the number of DCs that are generated for other grouping algorithms. K-means is used in our implementation. After clustering, the clusters are compared to DCs and each cluster is assigned to the most similar DC.) The recall of Pure Clustering is 0.75 compared to the recalls of CCC, QRW, Hybrid and E3C which are 0.89, 0.93, 0.94 and 0.95, respectively. The precision for all algorithms are greater or equal to 0.94.

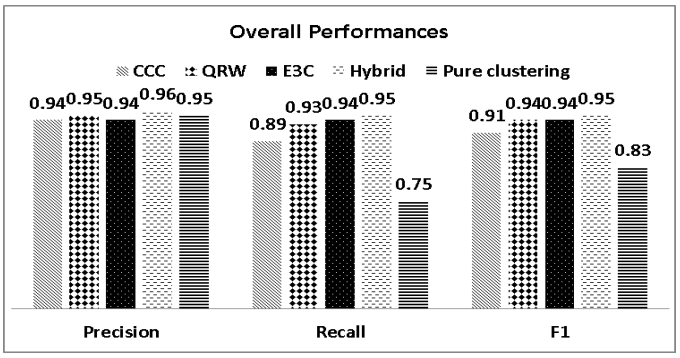

Fig. 3. Overall Performances

Each of the algorithms we have introduced in this paper has its own advantages and disadvantages. CCC, E3C and Pure Clustering are able to cover important web pages returned by a search engine since they all use the original query to retrieve results from a search engine. On the other hand, QRW may miss some important relevant results since we modified the original query and the modified query (QETs along with original query) is used to retrieve results for each specific DC from a search engine.

Furthermore, in CCC and E3C, we may not be able to find relevant SRRs for all DCs when only some top-ranked SRRs are retrieved per query (e.g., 50 SRRs). On average only 67% of the top 5 DCs can have assigned SRRs from the top 50 retrieved results in our dataset. This is due to the fact that those DCs were less popular and there were no relevant SRRs for those DCs among the top 50 results when the original queries were submitted. In QRW, we can find relevant SRRs for all DCs.

6 Conclusion

In this paper we investigated the problem of how to group the search result records (SRRs) from search engines. We previously have generated and ranked all possible categories of any user queries according to their match with the expected intention of the user. These categories can be used to categorize SRRs returned from search engines in response to user queries. By grouping the SRRs based on the different meanings/usages of the query term, it makes it easier for users to identify relevant results from the retrieved results. In this paper we focused on grouping SRRs and studied different grouping algorithms. Specifically, we proposed four algorithms that

combine categorization, clustering and query expansion techniques. Our novel grouping algorithms use three different sources (Wikipedia, WordNet and SRRs) fully automatically to categorize the SRRs in a unique way. The categories built by our method are more meaningful and distinguishable than those by existing techniques since we build our categories based on different meanings/usages of queries' terms. We cover all possible meanings and usages of any terms and phrases by using WordNet and Wikipedia. We can also cluster uncategorized SRRs for those SRRs that do not belong to any categories built based on Wikipedia and WordNet. Our experimental results indicated that our SRR grouping algorithms are effective and highly accurate.

References

1. R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In Proc. 2nd ACM Intl Conf on Web Search and Data Mining, 2009
2. C. Carpineto, S. Osinski, G. Romano, & D. Weiss, A survey of web clustering engines. ACM Computing Surveys, 41(3), Article No. 17, 2009
3. M. He, M. Cutler, K. Wu, Categorizing Queries by Topic Directory. WAIM Conference, pp.278-284. 2008
4. R. T. Hemayati, W. Meng, C. Yu. Semantic-based Grouping of Search Engine Results Using WordNet **.** APWeb/WAIM'07, pp.678-686, 2007
5. R. T. Hemayati, W. Meng & C. Yu. Identifying and Ranking Possible Semantic and Common Usage Categories of Search Engine Queries. WISE Conf., 2010
6. R. Hemayati, W. Meng, C. Yu. Categorizing Search Results. Technical report, http://cs.binghamton.edu/~rtaghiz1/, 2012
7. A. Hotho, S. Staab, G. Stumme. WordNet Improves Text Document Clustering. ACM SIGIR Semantic Web Workshop, 2003
8. A.K. Jain, M.N. Murty. Data Clustering: A Review. ACM Computing Surveys, 1999
9. F. Liu, C. Yu, W. Meng, Personalize Web Search by Mapping User Queries to Categories. ACM CIKM Conference, 2002
10. S. Liu, C. Yu, and W. Meng. Word Sense Disambiguation in Queries. ACM CIKM Conference, pp.525-532, 2005.
11. E. de Luca, A. Nürnberger, O. von-Guericke. Ontology-Based Semantic Online Classification of Documents: Supporting Users in Searching the Web. University of Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany, AMR, 2004
12. E. Pitler , K. Church, Using word-sense disambiguation methods to classify web queries by intent, Conference on Empirical Methods in NLP: Volume 3, 2009
13. R. L. T. Santos, C. Macdonald, and I. Ounis. Intentaware search result diversification. In SIGIR, 2011
14. T. de Simone and D. Kazakov. Using WordNet Similarity and Antonymy Relations to Aid Document Retrieval. Recent Advances in Natural Language Processing (RANLP), 2005
15. M.H. Song, S.Y. Lim, D.J. Kang, S.J. Lee. Ontology-Based Automatic Classification of Web Documents. ICIC (2) : 690-700. 2006
16. D.Xing, G. Xue, Q. Yang, Y. Yu, Deep Classifier: Automatically Categorizing Search Results into Large-scale Hierarchies. Int'l conf. on Web Search & Data Mining. 2008
17. O. Zamir, O. Etzioni, Grouper: A Dynamic Clustering Interface to Web Search Results. World Wide Web Conference, 1999
18. H. Zeng, Q. He, Z. Chen, and W. Ma. Learning To Cluster Web Search Results. ACM SIGIR, 2004.