Semantic-Based Grouping of Search Engine Results Using WordNet

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

[1] Department of Computer Science
State University of New York at Binghamton
Binghamton, NY 13902, USA
{rtaghiz1,meng}@binghamton.edu
[2] Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607, USA
yu@cs.uic.edu

Abstract. Terms used in search queries often have multiple meanings. Consequently, search results corresponding to different meanings may be retrieved, making identifying relevant results inconvenient and time-consuming. In this paper, we propose a new solution to address this issue. Our method groups the search results based on the different meanings of the query. It utilizes the semantic dictionary WordNet to determine the basic meanings or senses of each query term and similar senses are merged to improve grouping quality. Our grouping algorithm employs a combination of categorization and clustering techniques. Our experimental results indicate that our method can achieve high grouping accuracy.

Keywords: Categorization, Clustering, WordNet, Search Engine.

1 Introduction

Most Web users use search engines to find the information they want from the Web. One common complaint about the current search engines is that they return too many useless results for users' queries. Both the search engines and the users contribute to this problem. On the one hand, current search engines make little effort to understand users' intentions and they retrieve documents that match query words literally and syntactically. On the other hand, Internet users tend to submit very short queries (average length is about 2.3 terms and 30% have a single term [8]). One way to tackle this problem is to group the search results for a query into multiple categories such that all results in the same category corresponds to the same meaning of the query.

In this paper we propose a new technique to group the *search result records* (**SRR**s) returned from any search engine. Our focus will be on SRRs retrieved by *single term queries*. For queries with multiple terms, the specific meaning of each term is easier to determine because other terms in the same query can provide the context [9]. Our technique differs from existing techniques in the following aspects. First, we use a semantic electronic dictionary WordNet [5, 13] to provide the basic

G. Dong et al. (Eds.): APWeb/WAIM 2007, LNCS 4505, pp. 678–686, 2007.

meanings of each query term. Second, we apply a merging algorithm to merge synsets that have very close meanings into a super-synset. Third, we employ a two-step process to categorize SRRs into super-synsets. Fourth, our method also deals with SRRs that do not correspond to any WordNet-provided synsets of the query terms by clustering them. For example, when a word is used as a name, like "Apple" and "Jaguar", it does not have its traditional meanings. The rest of the paper is organized as follows. In section 2, we review some related work. In section 3, we describe our grouping algorithm. Preliminary evaluation results are reported in section 4. The future work and conclusion are presented in section 5.

2 Related Work

The general problem of document clustering and categorization has been studied extensively [7] 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 [14, 17, 18]. However, these techniques perform clustering based on the syntactical similarity but not semantic similarity. In contrast, our method employs both categorization and clustering techniques and it also utilizes similarities that are computed using both syntactical and semantic information.

Techniques for clustering and categorizing web documents using WordNet or other ontologies have also been extensively studied (e.g., [3, 4, 6, 12]) and some of them (e.g., [3, 4]) also tried to categorize SRRs based on the meanings of the query term. However, our approach differs from these techniques significantly. First, we use more features of WordNet such as hypernym, hyponym, synonym and domain. Second, we employ a sense-merging algorithm to merge similar senses before grouping. Third, our SRR grouping algorithm employs both categorization and clustering in a unique way. Fourth, our method also copes with SRRs that do not match any sense of the query term in WordNet. In other words, we utilize senses provided by WordNet but are not limited by them. Different techniques for clustering WordNet word senses are presented in [1] but they do not actually perform sense merging. These techniques can potentially be used for merging WordNet senses, e.g., merging the senses that are in the same cluster. However, our method is specific to the senses for the same query term, not for general sense merging. Consequently, our technique can be more efficiently applied to grouping SRRs. In addition, our merging algorithm is also different from the existing ones.

3 Grouping SRRs Using WordNet

Our method groups the SRRs not only based on their syntactically similar words but also semantically similar words. It is possible that two SRRs talk about very similar topics but they have less similar words while two other SRRs have more common words but they are less similar in reality. Since we compare the meanings and semantic relations between words, our method is more likely to yield better-grouped SRRs compared to current methods.

3.1 Method Overview

Our SRR grouping system for a user query *Q* consists of the following steps (Fig.1):

1. Send *Q* to a search engine/metasearch engine and process the returned SRRs.
2. While the query is being evaluated and the SRRs are being processed, send *Q* to WordNet to obtain its synonym sets (synsets) as well as terms that have certain semantic relationships with each synset.
3. Merge similar synsets into *super-synsets*.
4. Categorize SRRs by assigning each SRR to the most similar super-synset if the similarity is greater than a threshold T1. Temporary categories are obtained based on the current assignments and the remaining SRRs form another temporary category.
5. Further categorize the remaining SRRs by assigning each such SRR to the most similar temporary category.
6. Cluster the remaining SRRs.

We will explain each step in our algorithm in detail in the following subsections.

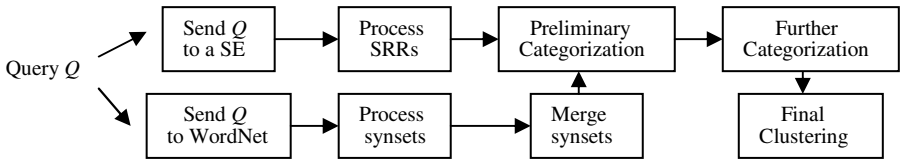

Fig. 1. Overview of our system

3.2 Submitting Query and Processing Results

For each user query, the top k (k = 50) distinct results (duplicates are removed) are retrieved and are used as input to our SRR grouping algorithm. 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. For each term, its term frequency (*tf*) in the SRR is recorded. The words in the title are considered to be more important than words in the snippet (we currently double the *tf* of each term in the title).

3.3 Sending Query to WordNet

Sending a user's query to WordNet means that certain information about the query term is obtained from WordNet and processed. This step is done in parallel to sending the query to the search engine and processing the returned SRRs. The fact that relationships between synsets are explicit is the motivation behind using WordNet in our approach. Synsets are linked using various types of relationship links. In our current approach, the following types of synsets are utilized for a given synset *S*:

- Hypernyms: Synsets that are more general in meaning than *S*
- Hyponyms: Synsets that are more specific in meaning than *S*

- Domains: Synsets that represent the domain of *S*
- Synonyms: Keywords that have the same meaning with the user query

This step (i.e., sending the query to WordNet) involves two procedures:

1. Get the senses/meanings for the query term. For each sense (synset), the synonyms, direct hypernyms/hyponyms, and the words in the definition and examples of the sense are all included in the representation of the synset. After removing stop words and stemming, each synset is represented as a vector of terms with weights that are computed from the term frequency of each term.
2. Merge similar senses if it's applicable. This step will be explained in section 3.4.

3.4 Sense Merging

It is often the case that some of the senses in WordNet fundamentally refer to the same concept or very similar concepts. The example below illustrates one such a case.

Example 1. Consider the following two synsets for query term "web"
Sense 1: web: (an intricate network suggesting something that was formed by weaving or interweaving; "the trees cast a delicate web of shadows over the lawn")

Sense 2: web, entanglement: (an intricate trap that entangles or ensnares its victim)

These two senses are very similar because both talk about physical webs with a subtle difference that the former emphasizes how the web is formed and the latter emphasizes how the web is used. The following is another sense of "web":

Sense 3: World Wide Web, WWW, web

It is easy to see that this sense is very different from the first two senses. The presence of synsets with similar meanings poses challenges to the SRR grouping algorithm as well as to the users who consume the grouped results. We propose to tackle this problem by merging the similar senses. Our sense-merging algorithm consists of five merging rules, each of which gives one condition under which two senses S1 and S2 can be merged. The five rules are given below:

Rule 1. If S1 and S2 have the same direct hypernym synset or one is a direct hypernym of the other, then merge S1 and S2.

Rule 2. If S1 and S2 have the same direct hyponym synset or one is a direct hyponym of the other, then merge S1 and S2.

Rule 3. If S1 and S2 have the same coordinate terms (i.e., there exist a synset S3 such that S1 and S3 share a direct hypernyn, and S2 and S3 also share a direct hypernym), then merge S1 and S2.

Rule 4. If S1 and S2 have common synonyms, then merge S1 and S2.

Rule 5. If S1 and S2 have the same direct domain synset or one is the domain of the other, then merge S1 and S2.

Intuitively, each condition in the above rules indicates that S1 and S2 are semantically similar.

3.5 Computing the Similarity Between SRRs and Super-Synsets

In this paper, we use a revised *Okapi function* to compute the similarity between SRRs and super-synsets. We made the changes to the original *Okapi function* [11] to fit our situation where two sets of documents are compared, one is the SRR set R* and the other is the super-synset set S*, whereas in the traditional information retrieval context, one document (the query) is compared with a set of documents. Our revised *Okapi function* for computing the similarity between an SRR R and a super-synset S is:

$$sim(R,S) = \sum_{T \in R \cap S} \frac{w_1 + w_2}{2} * w(T,R) * w(T,S) \quad (1)$$

with $w_i = \log \frac{N_i - n_i + 0.5}{n_i + 0.5}, i = 1,2$, $w(T,R) = \frac{(k+1) * tf(R)}{K(R) + tf(R)}$, and

$$K(R) = k * ((1-b) + b * \frac{dl(R)}{avgdl(R^*)})$$

where N_1 and N_2 are the numbers of SRRs in R* and super-synsets in S*, respectively, for the current query; n_1 and n_2 are the numbers of SRRs in R* and super-synsets in S* that contain term T, respectively; w_1 and w_2 reflect the importance of term T with respect to the SRRs in R* and the super-synsets in S*, respectively (they are similar to the *idf* weight in information retrieval); $w(T, R)$ computes the importance of term T in R; $tf(R)$ is the term frequency of T in R; $dl(R)$ is the length of R, and $avgdl(R^*)$ is the average length of all the SRRs in R^*; $k = 1.2$ and $b = 0.75$ are two constants. Finally, $w(T, S)$ is similar to $w(T, R)$ except R is replaced S and R^* is replaced by S^*.

3.6 SRR Grouping Algorithm

Our SRR grouping algorithm (Algorithm **CCC**) consists of the following three steps:

1. *Preliminary **C**ategorization*. Categorize SRRs based on their similarities with super-synsets. Specifically, for each SRR R, find the super-synset S that is most similar to R among all super-synsets. If sim(R, S) is greater than a threshold T1, then categorize R to S. At the end of this step, super-synsets with no SRRs categorized to them are removed. For each remaining super-synset S, its term vector and all the SRRs that are categorized into it are merged (i.e., the term sets are unioned and for each term, its term frequencies in S and the SRRs are added). Let the merged result be called the *expanded synset* from S.
2. *Further **C**ategorization*. Categorize the remaining SRRs from step 1 (i.e., those SRRs whose similarity with every super-synset is less than or equal to T1). Let RR be the set of remaining SRRs. Let R be an SRR in RR and C = RR – {R}. Find the expanded synset S that is most similar to R among all expanded synsets. If sim(R, S) > sim(R, C), then add R to the category corresponding to S and remove R from RR; otherwise, keep R in RR. When computing sim(R, C), the SRRs in C are temporarily merged.
3. *Final **C**lustering*. If there are still uncategorized SRRs left, we cluster them using a two-step clustering algorithm similar to the one in [10]. In the first step, use the

first SRR to form a cluster by itself and for each subsequent SRR *R*, place it in the most similar current cluster if the similarity is higher than a threshold T2; else create a new cluster based on *R*. This step is order sensitive and may leave some SRRs in less fitting clusters. In the second step, for each SRR *R*, we compute its similarity with the centroid of each cluster and move it to the cluster whose centroid is the most similar to *R* if *R* was not in this cluster. This is repeated until no *R* can be moved. *Cosine* similarity function is used in SRR clustering.

In our current implementation, the two thresholds T1 and T2 are determined using a training set. When training T1, we try to find the value that achieves the maximum possible *recall* under the condition that nearly every categorized SRR is assigned to the correct synset (i.e., close to 100% *precision*). In step 1, we try to be more conservative since we will have another chance to categorize the remaining SRRs in step 2. Consequently, after the SRRs categorized into a synset are merged at the end of step 1, each category is as accurately represented as possible. In step 2, the cluster *C* is considered because we want each remaining SRR to have a fair chance to be categorized or stay uncategorized, as it is possible that some SRRs do not match any senses from the WordNet. Step 3 is needed because many English words have non-standard uses in practice (such as used as a name of a company) that do not match any senses the WordNet has about these words and a query term may be a non-standard English word (such as "allinone").

4 Evaluation

We implemented our algorithm using Java. We use JWNL to connect to WordNet 2.0. Two datasets are used in this paper and each dataset contains 10 single-term queries and the 500 SRRs (50 unique SRRs per query) from search engine Yahoo. The 10 queries for the first dataset DS1 are (notebook, jaguar, mouse, metabolism, piracy, suicide, magnetism, web, people, salmon), and the 10 queries for the second dataset DS2 are (apple, dish, trademark, map, music, car, game, tie, poker, mold). DS1 is used for training to obtain the thresholds (T1 = 4 and T2 = 0.1 are obtained).

4.1 Alternative Solutions

As mentioned before for some terms, there are categories that are not covered in WordNet. For example, for query "jaguar", there are two categories not in WordNet, the first is the brand name for car and the second is unknown (some company names).

For our evaluation, we also compare the SRR grouping algorithm described in section 3.6 with two other intuitively reasonable solutions. Basically, each of the two alternative solutions replaces the last two steps (*Further categorization* and *Final clustering*) while the first step (*Preliminary categorization*) remains the same. In WordNet, a *frequency of use* is associated with each sense of a word and this value indicates how widely this sense of the word is used relative to other senses of the word. Our first alternative solution is based on the frequency of use. Note that during sense merging, the frequency of use of a super-synset is computed as the sum of the frequencies of use of all the individual synsets it contains.

- *Largest frequency of use* (**LF**): Assign all remaining SRRs (after the *Preliminary categorization* step) to the super-synset that has the largest frequency of use.

The rationale for this method is that the super-synset with the largest frequency of use represents the most common sense of the term among those covered by WordNet.

Our second alternative solution is based on the intuition that if, after the preliminary categorization step, a category has the most SRRs, then this category is probably the most popular category for the retrieved SRRs.

- *Largest category* (**LC**): Assign the remaining SRRs to the category that has the largest number of SRRs after the preliminary categorization step.

4 2 Performance Measures

We evaluate the sense-merging algorithm as well as the three SRR grouping algorithms (CCC, LF and LC). For all algorithms, we use the *recall*, *precision* and *F1 measure* (which combines recall and precision) as the performance measures. For the merging algorithm, we define precision = |A∩B|/|B| and recall = |A∩B|/|A|, where A is the set of merges that should be performed as judged by a human expert and B is the set of merges our merging algorithm performed. All the 20 queries in both datasets are used. Note that our merging algorithm does not need any training. For the SRR grouping algorithms, the recall and precision are defined below [10]:

- *Precision p*: For a given category, the precision is the ratio of the number of SRRs that are correctly grouped over the number of SRRs that are assigned to the group. For example, if among the 5 SRRs assigned to a group, only 4 are correct, then the precision for this group is 4/5 or 80%. The overall precision for all groups is the average of the precisions for all groups weighted by the size of each group. Specifically, the formula for computing the overall precision is

$$p = \sum_{i=1}^{n} p_i * \frac{N_i}{N}$$

  where p_i is the precision of the i-th group, N_i is the number of SRRs in the i-th group, N is the total number of SRRs (= 50) and n is the total number of groups.
- *Recall r*: For a given group, recall is the ratio of the number of SRRs that are correctly grouped over the number of SRRs that should have been grouped. For example, if a group should have 5 SRRs but an algorithm puts only 3 of them into this group, then the recall for this group is 3/5 or 60%. The overall recall for all clusters is the average of the recalls for all groups weighted by the size of each group. The formula for computing the overall recall is:

$$r = \sum_{i=1}^{n} r_i * \frac{N_i}{N}$$

  where r_i is the recall of the i-th group, and N_i, N and n are the same as in the definition of precision. When evaluating the SRR grouping algorithms, the final precision and recall are averaged over all queries.

Once precision p and recall r are computed, the *F1 measure* can be computed by $2*p*r / (p + r)$. The *F1 measure* is high only when both precision and recall are high.

4.3 Experimental Results

Our sense merging algorithm has a precision of 100, recall of 66 and F1-measure of 80. The results show that all merged senses are correct, but our algorithm still couldn't find all possible merges. Tables 1 and 3 show the results for the three SRR grouping algorithms based on DS1 and DS2 when merged senses are used. It can be seen that Algorithm CCC performs significantly better than Algorithms LF and LC. This is mainly due to the fact that the former can group the SRRs beyond the synsets in WordNet while the latter two methods force the SRRs that do not match any WordNet senses into incorrect categories. Another observation is that the results for the testing dataset DS2 are only slightly lower than the results for the training set DS1, indicating that the trained thresholds are reasonably robust. Tables 2 and 4 show the results when un-merged senses are used. It can be seen that sense merging helped the performance improve by approximately 5 percentage points. One of the reasons that causes incorrect grouping is the lack of common terms between some SRRs and the correct synset representations. We plan to investigate this problem in the future.

Table 1. With merged senses for DS1

Algorithm	Precision	Recall	F1
CCC	93%	91%	92%
LF	75%	77%	76%
LC	78%	80%	79%

Table 2. Without merged senses for DS1

Algorithm	Precision	Recall	F1
CCC	89%	86%	87%
LF	68%	70%	70%
LC	73%	74%	73%

Table 3. With merged senses for DS2

Algorithm	Precision	Recall	F1
CCC	90%	89%	89%
LF	74%	77%	75%
LC	77%	79%	78%

Table 4. Without merged senses for DS2

Algorithm	Precision	Recall	F1
CCC	85%	83%	84%
LF	65%	67%	66%
LC	69%	70%	69%

5 Conclusions and Future Work

In this paper, we investigated the problem of how to group the search result records from search engines (or metasearch engines) for single-term queries. Single-term queries are often ambiguous because many English words have multiple meanings. By grouping the search results based on the different meanings of the query term, it makes it easier for users to identify relevant results from the retrieved results. We proposed a novel three-step grouping algorithm that combines both categorization and clustering techniques. We also proposed an algorithm to merge similar senses returned from WordNet. Our preliminary experimental results indicated that our SRR grouping algorithm is effective, achieving an accuracy of about 90%. We also showed that this algorithm is significantly better than two other possible solutions and our sense-merging algorithm can improve grouping accuracy by about 5%.

We plan to continue this research in the following directions. First, we plan to conduct more experiments using a significantly larger dataset. Second, we will try to

improve our sense-merging algorithm and SRR grouping algorithm as there are still rooms for improvement. Third, while WordNet is very useful, it is far from perfect in providing all the senses for many words. We plan to see if other online semantic dictionaries, such as Wikipedia, can also be utilized. Finally, we also plan to develop good SRR grouping solutions for multi-term queries.

Acknowledgment. This work is supported in part by the following NSF grants: IIS-0414981, IIS-0414939 and CNS-0454298.

References

1. E. Agirre, E. Alfonseca, O. Lopez. Approximating Hierarchy-based Similarity for WordNet Nominal Synsets Using Topic Signatures. Global WordNet Conference, 2004.
2. G. Attardi, A. Cisternino, F. Formica, M. Simi, A. Tommasi. PiQASso 2002. TREC11 2002.
3. E. W. De Luca and 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.
4. T. de Simone and D. Kazakov. Using WordNet Similarity and Antonymy Relations to Aid Document Retrieval. Recent Advances in Natural Language Processing (RANLP), 2005.
5. C. Fellbaum (edited). WordNet: An Electronic Lexical Database (Language, Speech & Communication). The MIT Press, 1998.
6. A. Hotho, S. Staab, G. Stumme. WordNet Improves Text Document Clustering. ACM SIGIR Semantic Web Workshop, 2003.
7. A.K. Jain, M.N. Murty. Data Clustering: A Review. ACM Computing Surveys, 1999.
8. B. Jansen, B. Spink, J. Bateman, T. Saraceric. Real Life Information Retrieval: A Study of User Queries of the Web. ACM SIGIR Forum, 32(1), pp.5-17, 1998.
9. S. Liu, C. Yu, and W. Meng. Word Sense Disambiguation in Queries. CIKM, 2005.
10. Q. Peng, W. Meng, H. He, and C. Yu. WISE-Cluster: Clustering E-Commerce Search Engines Automatically. WIDM, 2004.
11. S. Robertson, S. Walker, M. Beaulieu. Okapi at Trec-7: Automatic Ad Hoc, Filtering, Vlc, and Interactive Track. 7th Text REtrieval Conference, 1999, pp.253-264.
12. M.H. Song, S.Y. Lim, D.J. Kang, S.J. Lee. Ontology-Based Automatic Classification of Web Documents. ICIC (2) 2006: 690-700.
13. WordNet; http://wordnet.princeton.edu/
14. Vivisimo, http://www.vivisimo.com
15. Y. Yang, S. Slattery and R. Ghani. A Study of Approaches to Hypertext Categorization, Journal of Intelligent Information Systems, 18(2), March 2002.
16. Y. Yang. A Study of Thresholding Strategies for Text Categorization. ACM SIGIR, 2001.
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.