An Unsupervised Approach to Cluster Web Search Results based on Word Sense Communities

Jiyang Chen, Osmar R. Zaïane and Randy Goebel
Department of Computing Science
University of Alberta, Canada T6G 2E8
{jiyang, zaiane, goebel}@cs.ualberta.ca

Abstract

Effectively organizing web search results into clusters is important to facilitate quick user navigation to relevant documents. Previous methods may rely on a training process and do not provide a measure for whether page clustering is actually required. In this paper, we reformalize the clustering problem as a word sense discovery problem. Given a query and a list of result pages, our unsupervised method detects word sense communities in the extracted keyword network. The documents are assigned to several refined word sense communities to form clusters. We use the modularity score of the discovered keyword community structure to measure page clustering necessity. Experimental results verify our method's feasibility and effectiveness.

1 Introduction

Existing search engines often return a long list of search results, ranked by their relevance to the given query. Since web pages, on different aspects (meanings) of the same query, are usually mixed together, users have to go through the long list and examine titles and content sequentially to locate pages of interest to their information need. For example, when the query "jaguar" is submitted to a search engine looking for information about *the Mac system*, the user might have to sift through a large number of pages about *automobile* or *animals*. The sought for pages might be buried very deep. While the underlying retrieval model and ranking function is vital for search engines, organization and presentation of search results is also capital, and could significantly affect the utility of a search engine. However, compared with the vast literature on page ranking and retrieval, there is relatively very little research on how to improve the effectiveness of search result organization [14].

A possible solution to this problem is to (online) cluster search results into different groups so that users can select their required group at a glance. Previous approaches to document grouping usually require high quality training data to build a classifier, which is infeasible due to the dynamic nature of the web and the need of a realtime answer. Some clustering approaches for search engine results exist. However, since clustering methods always generate groups of documents, even when unnecessary, a measure is required to indicate when page clustering is helpful. In this paper, we apply community detection ideas on the problem of page clustering based on discovered senses of a given query. Our work has the following contributions:

- An unsupervised method to identify query senses and cluster web pages using a community mining approach on a network of extracted keywords.
- The use of the modularity Q measure of the discovered word sense community structure to assess whether page clustering is required for search results.

The rest of the paper is organized as follows. We discuss related work in Section 2. Section 3 introduces the word sense community definition. We describe our approach in four phases in Section 4 and report experimental results in Section 5, followed by conclusions in Section 6.

2 Related Work

In general, there are two ways to organize the information returned by search engines. The first, and the most popular approach, is to rank results by perceived relevance. However, this method is highly inefficient since there are usually thousands of retrieved pages for a typical query, and most users just view the few top results, possibly missing relevant information. Additionally, a query might have different meanings. The inherent ambiguity in interpreting a word or phrase in the absence of its context means that a large percentage of the returned results can be irrelevant to the user [4]. The second way to organize documents is by *clustering*: query result pages are organized into groups based on their similarity between each other. The idea of clustering search results has already been applied in industry and commercial web services such as Vivisimo [13], Kartoo [3] and Koru [7]. Zamir et al. [16] proposed to cluster query result pages based on the snippets or contents of returned documents using the Suffix Tree Clustering (STC) algorithm. In [4], the authors proposed a monothetic clustering algorithm to assign documents to clusters based on a single feature, which is used as cluster labels. Zeng et al. [17] proposed a supervised learning approach to extract relevant phrases from the query result snippets, which are used to group search results. Wang et al. [14] proposed an approach to learn from search logs for a more user-oriented partitioning of the search results. All these methods are able to find document clusters but the effectiveness in practical search engine result partitioning is questionable. In addition, the performance of supervised classifiers is limited by the training data, which is hard to achieve for the dynamic web. Moreover, none of the proposed methods measures clustering necessity (i.e. document clustering is not necessary if the original search result contains no ambiguity).

3 Detecting Word Sense Communities

The *Contextual Hypothesis for Sense* states that the context in which a word appears is usually related to its sense [10]. For example, a web page discussing Jaguar as a *car* is likely to talk about other types of cars, car companies, etc. Whereas, a page on Jaguar the *cat*, is likely to contain information about other kinds of animals. Naturally, the words or phrases that frequently appear together with Jaguar the car, e.g., *engine, Ford* and *vehicle*, are very unlikely to also appear frequently in web pages about Jaguar the cat, and vice versa. Moreover, there are extremely few pages that discuss both senses of Jaguar in detail at the same time. Therefore, the sense of the word Jaguar and other words that frequently appear together with this particular sense can be used to cluster search result pages for this particular query "jaguar", which is the intuition of our approach.

We define a word sense community as *a group of keywords or phrases that co-appear frequently in a set of search result pages for a particular query*. There are several possible definitions for "keyword co-appearance", e.g., within a given distance d, in the same paragraph, etc. In this paper, we define two words to "co-appear" if they are located in the same sentence. Consider a graph, where each node represents a keyword/phrase and the edge between two nodes represents their co-appearance, the edge weight is the frequency of the two keywords co-occurring in one sentence, we extend the Modularity metric [8] to its weighted version and then adapt a hierarchical algorithm [1] on this keyword graph to detect word sense communities.

4 Our Approach

We use the word sense communities discovered by the modularity approach on our word/phrase network to cluster search results. Given an input query, the general procedure of our approach can be described in four phases.

4.1 Keyword Extraction

The first step to extract word sense communities from web pages is to build the keyword network, weighted by the frequency of the sentence co-appearance between keywords. Given a query q, we send q to the Google search engine and retrieve the top k returned web pages. We parse the content of text pages (such as pages ending with *.html*, *.php*, etc.) and ignore multimedia pages. Irrelevant information such as HTML tags and javascript code is stripped and only text is recovered.

We use Minipar [5], a broad-coverage English parser, to parse the clean text. Minipar is able to transform a complete sentence into a dependency tree and classify words and phrases into lexical categories. We only use nouns as keywords in our approach and ignore other lexical categories such as verbs, adjectives, adverbs and pronouns, since they can be used in various contexts thus usually do not belong to one particular word sense community. All retrieved keywords are then stemmed using the Porter Stemming Algorithm [12] and stopwords are removed. Note that, although keyword extraction is slow since Minipar parsing is time-consuming (500 words/second on a normal PC [9]), it should be done offline during web page crawling by the search engine crawler, therefore the running time of our approach during query time is not affected.

After text parsing and stemming, each page is represented as a list of pairs of keywords, which have been located in a same sentence. Assume we have k such lists representing k result pages for query q, in order to find important keywords in these documents to build the keyword network, we measure the importance of keywords by the Inverse Document Frequency (IDF), which is calculated by dividing the number of all documents by the number of documents containing the keywords. We then select those keywords that have IDF score higher than a given threshold t_{idf} to be nodes of the keyword network. Two nodes are connected if we find a pair of corresponding keywords in the lists and the weight of the edge is the pair frequency. The query words are removed since they certainly belong to all sense communities.

4.2 Find Word Sense Communities

In order to find word sense communities from the weighted keyword network, we extend the modularity to its weighted version. Given an undirected network $G = (V, E)$, $|V| = n, |E| = m$, let A_{xy} be an element of the adjacency matrix of G.

$$A_{xy} = \begin{cases} w & \textit{if vertices x and y are connected} \\ 0 & otherwise \end{cases}$$

where w is the edge weight. Also, $P_{xy} = \frac{w_x w_y}{2W}$ where w_x is the total weight of all edges connect to x and W is the total weight of the network. $Q_{weighted}$ equals to:

$$Q_{weighted} = \frac{1}{2W} \sum_{xy} [A_{xy} - P_{xy}] \phi(C_x, C_y)$$

Assume node x belongs to community C_x, the ϕ function $\phi(C_x, C_y)$ is 1 if C_x and C_y are the same community and 0 otherwise. See [1] for details of Q transformation.

Any modularity-based clustering algorithm could be applied here. In this paper, we adapt a hierarchical clustering algorithm to greedily optimize the modularity score [1]. It starts as every node being a community of its own, then at each step, it merges a pair of communities that increase the overall modularity the most and stops when there is no such pair. Since a high modularity score represents strong community structure, the intuition of this algorithm is to greedily optimize the overall modularity. Details of the algorithms are as follows: Given a weighted keyword network, three data structures are maintained.

- A sparse matrix containing ΔQ_{ij} for each pair i, j of communities with at least one edge between them. Each row of the matrix is stored as a balanced binary tree and as a max heap. The matrix is initialized as:

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2W} - \frac{w_x w_y}{(2W)^2} & \textit{if i and j are connected} \\ 0 & otherwise \end{cases}$$

- A max-heap H containing the largest element of each row of the matrix along with the labels i, j of the corresponding communities.
- A vector array with element $a_i = \frac{w_i}{2W}$

The algorithm then greedily merges pairs of communities that give the highest modularity gain as follows:

- Pop the max-heap with the largest element of each row of the matrix ΔQ.
- Select the largest ΔQ_{ij}, merge the two communities, update ΔQ (described blow), the heap H and a_j ($a'_j = a_i + a_j$), increment Q by ΔQ_{ij}.
- Repeat until there is no $\Delta Q_{ij} > 0$

Merging community i and j by updating ΔQ as follows.

$$\Delta Q'_{jk} = \begin{cases} \Delta Q_{ik} + \Delta Q_{jk} & \textit{if community k is connected to} \\ & \textit{both i and j} \\ \Delta Q_{ik} - 2a_j a_k & \textit{k is connected to i but not to j} \\ \Delta Q_{jk} - 2a_i a_k & \textit{k is connected to j but not to i} \end{cases}$$

4.3 Community Refinement

After we have discovered word sense communities from the weighted keyword network, we refine the structure for the following two situations:

- Delete noise communities, which are formed by keywords that always co-appear no matter what the page is about, e.g., we observe keywords *trademark*, *privacy* and *policy* always form a strong community together.
- Merge communities that share the same word sense but focus on different aspects of the sense, e.g., we observe two communities for the sense of *Java* as *the programming language*, one focus on how to program, the other represents topics on the Sun company and its business.

Fortunately, simple heuristics can be applied to solve the problem. We observe that noise communities are usually small in size. Therefore, we remove all communities that have fewer nodes than 5% of the total keywords (note that this threshold is stable enough. Varying it from 5% to 10% does not affect the result). For merging communities, recall that we assume that there is only one primary word sense for a given query in one web page, thus if two communities share the same word sense, they are likely to be covered by the same page. Therefore, we calculate the overall TF-IDF score (described in Section 4.4) of pages for these two communities, and compare the two sets of pages whose scores exceed a threshold t_{merge} . If the size of overlapping pages is more than half of one of the page set, we merge the two communities in question. These heuristics work well as shown by our experiments in Section 5.

4.4 Assign Documents to Labeled Communities

Our final step is to assign pages to communities and label them. In order to assign a page p to its most related word sense community, we calculate the overall TF-IDF score of p for all communities and assign p to the one that has the highest score. If more than one candidate community has the highest score, we categorize p as *miscellaneous*. The overall TF-IDF score of p for community c is defined as the sum of TF-IDF scores of all keywords, which belong to community c, for p.

We use the dependency-based word similarity data[1] [6] to label the clusters. For a keyword w in community c, we sum w's similarity ranking to all other keywords in c as an overall ranking for w. We use keywords that have high overall ranking as c's label. More accurate document cluster labeling methods are possible to apply here. However, cluster labeling itself is a huge research topic and thus is beyond the scope of this paper.

[1]http://www.cs.ualberta.ca/∼lindek/downloads.htm

DataSet	Manual Label	Dependency-based Keyword	ARI score			Q score
			Our Method	K-means	Human h	
Amazon	River	lake, river, water, ocean, forest	0.888	0.693	1	0.367
	Warrior	girl, battle, woman, artist, writer				
	Company	computer, consumer, rate, database				
Java	Coffee	coffee, fruit, tea, vegetable, sugar	0.889	0.728	0.964	0.403
	Island	island, mountain, city, coast, resort				
	Software	software, interface, graphic, application				
Eclipse	Car	engine, car, video, audio, vehicle	0.931	0.765	0.955	0.428
	Solar	sun, picture, moon, earth, light				
	Java	software, interface, server, application				
Jaguar	Animal	animal, wildlife, forest, tiger, bird	0.785	0.114	0.961	0.471
	Car	car, vehicle, truck engine, sedan				
	Mac	database, software, interface, file, server				
Salsa	Dance	music, dance, teacher, jazz, musician	0.642	0.605	0.974	0.405
	Sauce	garlic, tomato, onion, sauce, lemon				
Reuter	Trade	budget, tax, tariff, export, import	0.618	0.504	1	0.222
	Crude	oil, crude, supply, price, output				
	Money-fx	currency, market, dollar, rate, franc				

Table 1. Sense community-based clusters for six datasets (miscellaneous clusters are omitted).

Dataset	Manual Labels	Page Set Size
amazon	river, warrior, company	114
java	software, island, coffee	119
eclipse	car, solar, java	125
jaguar	car, animal, mac	101
salsa	dance, sauce	85
Reuters*	Trade, Crude, Money-fx	946

Table 2. Experimental Datasets

5 Experiment Results

We constructed our datasets using the Google search engine and partitioned the results manually to create ground truth. At first, we submitted ambiguous queries to Google and parsed top returned page results. Pages that do not contain any keyword pairs were removed. We merged result pages from several related queries to create datasets that have strong word sense communities on purpose, e.g., the *amazon* dataset is merged by results from queries "amazon river", "amazon warrior" and "amazon company", and so is the *java* (island, coffee or programming language) and *eclipse* (Mitsubishi car model, obscuring of a celestial body, or programming development platform) datasets. For *jaguar* and *salsa* datasets, we queried the word "jaguar" and "salsa" only. Table 2 lists the labeled datasets. In order to build ground truth to evaluate our results, we asked four graduate students to manually classify all pages into predefined clusters using a vote system, i.e., a page is classified to the cluster which most people agree on. If votes are even, we have a fifth person to make the final decision[2]. Pages can also be labeled as *miscellaneous* if they do not belong to any of the pre-defined clusters.

For practicality, we kept the sets small to be manually labeled. However, to test on a larger set, we used a subset of the standard text data set Reuters-21578[3] and selected three document categories with about the same size, to simulate a query with three senses. (see Table 2). Totalling about 950 documents, each is treated as a parsed page.

5.1 Overall Performance

We have applied our method on the six datasets and show the results in Table 1 (miscellaneous clusters are omitted). We set $t_{idf} = 0.05$ and $t_{merge} = 0.2$ (details are omitted due to lack of space). To evaluate how closely each community in the result matches its corresponding community in ground truth, we adopt the Adjusted Rand Index (ARI) [15] as the performance metric for accuracy. We compare our method with an effective variation of K-Means [2], which is a common algorithm of document clustering [11], and the labeling by one student (human h). For the K-Means algorithm, every extracted keyword is treated as a feature, thus one document is represented as a vector of keyword TF-IDF scores. The distance between two documents is defined as the squared Euclidean distance between two vectors. From the table, we see that for datasets (*amazon, java, eclipse*) that are merged by three different sense of the same

[2]http://www.cs.ualberta.ca/∼jiyang/WI2008/.
[3]http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

Specified Query	Modularity Score Q
amazon river	0.269
amazon warrior	0.226
java coffee	0.155
java software	0.182
solar eclipse	0.268
eclipse java	0.202

Table 3. Modularity for different queries

query, our approach achieves high accuracy, which validate our assumption that a word sense community relates to its corresponding document cluster. For real datasets (*jaguar*, *salsa*, *reuter*) with noise, our method still works measurably well and detects the right number of clusters. Note that we use the cluster number discovered by our approach as k to feed the k-means algorithm. While our approach detects k automatically based on senses of query words, other unsupervised algorithms often rely on such critical parameter, thus our approach is more appropriate for real time search result page clustering, where such information is unavailable. Also note that our running time depends on the size of the *keyword network*, thus we are able to handle large document sets very fast since the number of extracted keywords is stable regardless of the number of pages.

5.2 Using Q to Measure need for Partitioning

Obviously, a list of result pages does not need clustering if it only contains one primary sense of the query. The stronger its sense community structure is, the more confusing the result could be. This can be measured by the Q score, since Q indicates the strength of the community structure: Q is close to 0 if there is only one sense and higher score means a page clustering is necessary. Typically, $Q \geq 0.3$ indicates strong community structure[8].

In Table 1 and 3, we show Q scores for detected sense community structure of various queries. For queries that are indiscriminate, such as *amazon*, *java*, *eclipse*, *jaguar and salsa*, Q scores for the sense community structure of corresponding result pages are high compared to scores for more specified queries in Table 3. For instance *java coffee* need not be partitioned given the low Q score. Note that some Q of specified queries are not low since there are still small-scale communities inside, e.g., for *solar eclipse*, we find that there are two main communities, one of which focuses on scientific analysis and the other discusses photography.

6 Conclusions

This paper proposed an approach for web page clustering based on word sense community detection in documents. Our unsupervised method bypasses the problem of handling large result page sets by extracting and analyzing important keywords and phrases only, also it is able to achieve high clustering accuracy for real queries. The modularity score Q can be used to measure whether a page clustering is required. Experimental results confirm the accuracy and effectiveness of the proposed approach. Possible future work would be extending the approach to build a document hierarchy based on merged topics and discovered senses.

References

[1] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very lage networks. *Phys. Rev. E*, 70:066111, 2004.

[2] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, 2002.

[3] Kartoo. http://www.kartoo.com/.

[4] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *WWW*, pages 658–665, 2004.

[5] D. Lin. Principar: an efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th conference on Computational linguistics*, pages 482–488, 1994.

[6] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, 1998.

[7] D. N. Milne, I. H. Witten, and D. M. Nichols. A knowledge-based search engine powered by wikipedia. In *CIKM*, pages 445–454, 2007.

[8] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, 2004.

[9] P. Pantel and D. Lin. Discovering word senses from text. In *KDD*, pages 613–619, 2002.

[10] H. Schütze. Automatic word sense discrimination. *Comput. Linguist.*, 24(1):97–123, 1998.

[11] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *In Proceedings of Workshop on Text Mining, KDD'00*, pages 109–110, 2000.

[12] P. Stemming. http://tartarus.org/ martin/PorterStemmer/.

[13] Vivisimo. http://www.vivisimo.com/.

[14] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *SIGIR*, pages 87–94, 2007.

[15] K. Y. Yip and M. K. Ng. Harp: A practical projected clustering algorithm. *IEEE TKDE*, 16(11):1387–1397, 2004.

[16] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[17] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR '04*, pages 210–217, 2004.