Exploiting User Queries for Search Result Clustering

Abdul Wahid, Xiaoying Gao, and Peter Andreae

School of Engineering and Computer Science,
Victoria University of Wellington,
19 Kelburn Parade 6012. Wellington, New Zealand
{abdul.wahid,xgao,pondy}@ecs.vuw.ac.nz
http://ecs.victoria.ac.nz

Abstract. Search Result Clustering (SRC) groups the results of a user query in such a way that each cluster represents a set of related results. To be useful to the user, the different cluster should contain the results corresponding to different possible meanings of the user query and the cluster labels should reflect these meanings. However, existing SRC algorithms often ignore the user query and group the results based just on the similarity of search results. This can lead to two problems: *low quality cluster*, where the results within a single cluster are related to different meanings of the query; and *poor cluster labels*, where the label of the cluster does not reflect the query meaning associated with the results in the cluster.
This paper presents a new SRC algorithm called QSC that exploits the user query and uses both syntactic and semantic features of the search results to construct clusters and labels. Experiments show that the query senses are good candidates for the cluster labels and the algorithm can lead to high quality cluster and more semantically meaningful labels than other state-of-the-art algorithms.

Keywords: Web Clustering Engine, Search Result Clustering, Query Senses, Document Clustering

1 Introduction

The most important job of search engines is to display the results of a user query in such a way that user can easily find the desired information. A user query that has multiple senses is often called an ambiguous query. The user query *jaguar* is an ambiguous query e.g. it can mean American big cat, the British car manufacturer, an operating system by Apple, etc. Traditional search engines will return the list of ranked pages without considering the different senses of the query, leaving the user to browse through irrelevant pages. In contrast, clustering search engines such as Carrot use Search Result Clustering algorithms that group documents and provide descriptions for these groups enabling users to find their desired results in a more efficient way.

The goal of Search Result Clustering is not only to cluster search results but also to provide semantically meaningful cluster labels. A Cluster label is a one-phrase description of all the documents in a cluster enabling users to decide whether to browse the list of documents in a cluster by looking at the cluster label. It is a common practice to use the most common keywords shared by all the documents in a cluster as a cluster label. Documents can have common keywords that might represent either more than one sense or might not represent any sense of the user query. Therefore, cluster labels based on common keywords are not always useful to the user. Also the clusters will be more useful to the user if all the documents in a cluster represent only one particular sense of the user query.

Traditional Search Result Clustering algorithms which ignore the user query are more vulnerable to the problems of *low quality cluster* and *poor cluster labels*. *Low quality cluster* is having documents in a cluster that represent more than one senses of the user query and *poor cluster labels* are cluster labels that do not represent any senses of the user query.

The similarity between two documents is often measured using word frequency. Such similarity measures are regarded as syntactic measures because they only consider counts of words. In order to minimize the problem of *low quality cluster*, this paper uses both syntactic and semantic features (topics) of the documents.

This paper presents a new algorithm Query Sense Clustering (QSC) that exploits the user query and combines semantic and syntactic features of a document for the clustering solution. The paper is organized as follows: section 2 highlights the related work; section 3 discusses the representation and similarity measures of the documents and the query senses; section 4 describes the algorithm; section 5 focuses on the evaluation and analysis of the results and section 6 concludes the paper.

2 Related Work

Search Result Clustering (SRC) methods can be classified into three categories: data-centric, description-aware and description-centric [5].

The data-centric category contains traditional clustering algorithms (hierarchical, partitioning) and the focus is on the clustering process. The Scatter/Gather algorithm [15, 29] is the pioneer example of the data-centric category. Other prominent examples are WebCat [14], AISearch [32], LASSI [19], TRSC [25] and Link-based clustering [38]. The main drawback of this category is the *poor cluster labels* which are often generated from the text and are often meaningless.

The description-aware methods carefully select one or more features to construct meaningful cluster labels. Suffix Tree Clustering (STC) [36, 35] was the first algorithm that used suffix trees to build cluster labels and perform clustering on search results. Later it was improved by fixing the scalability issue [4, 20], implementing better scoring function (ESTC) [8] and approximating sentences

for better document similarity (SnakeT) [13]. The issue with description-aware methods is that the cluster labeling procedure dominates the clustering process and the overall quality of the clusters is compromised.

The description-centric methods are specialized clustering methods that not only focus on cluster labels but also try to provide quality clusters. Examples in this category include LINGO [26], DisCover [18], CREDO [6], KeySRC [1] and STHAC [34]. Our algorithm QSC also belongs to this category.

The user query was first used by QDC [9] to guide the clustering process. Later EQDC [28] improved the QDC method to handle key phrases. However it was based on ESTC and does not consider the query senses explicitly.

Apart from the above, word sense induction based methods such as Curvature [12], SquaT++ [24, 11], B-MST [10], HyperLex [33], Chinese Whispers [2] are related to this paper because they use query senses. They all identify multiple senses of the query by applying graph based word co-occurrence techniques. This paper identifies query senses from Wikipedia and is more similar to SRCluster [22]. However both word sense induction based methods and SRCluster use syntactic features of the document whereas in this paper we use both syntactic and semantic features of the document to construct the similarity measure. The comparison results of our algorithm with these algorithms are given in section 5.

3 Representation and Similarity Measure

This work uses query senses to generate initial clusters and then uses a new document similarity measure to refine the initial clusters. The new document similarity measure is based on a new document representation using both syntactic and semantic features (topics). The following subsections introduce the new document representation, the document similarity measure, the query sense representation and the sense similarity measure. The algorithm is presented in section 4.

3.1 Document Representation

The traditional bag-of-words model is widely used in document clustering to represent documents in Vector Space. Terms are commonly weighted using the tf-idf weighting scheme [31]. A document d in term-space is represented as

$$Tm(d) = \{tfidf(t_1, d), tfidf(t_2, d), tfidf(t_3, d), ..., tfidf(t_n, d)\} \quad (1)$$

where n is the total number of terms and $tfidf$ is the tf-idf function defined as

$$tfidf(t, d) = tf(t, d) \times \log \frac{|D|}{df(t)} \quad (2)$$

where $tf(t, d)$ is the frequency of term t in the document d, $|D|$ is the total number of documents and $df(t)$ is the number of documents containing term t.

A criticism of this model is that it only uses a syntactic representation of the document and ignores semantic representation of the document. One semantic representation is based on topics representing the subjects or concepts that a document is about. If we can identify all the topics of a documents, then we can represent a document as a vector in topic space with weights for each topic representing the importance of the topic to the document. We propose a new document representation in which a document d containing topics $\tau_1 ... \tau_m$ in topic-space is represented as

$$Tp(d) = \{w(\tau_1, d), w(\tau_2, d), w(\tau_3, d), ..., w(\tau_m, d)\} \tag{3}$$

where m is total number of topics and $w(\tau, d)$ is a weight of a topic τ, generated using topic detector of Wikiminer Toolkit [23], in document d.

3.2 Document Similarity Measure

The most common and well known similarity measure for comparing documents is cosine similarity function [27]. We define the combined cosine similarity that includes semantic and syntactic features of document d_i and d_j as

$$Sim(d_i, d_j) = \lambda Cosine(Tp(d_i), Tp(d_j)) + (1 - \lambda) Cosine(Tm(d_i), Tm(d_j)) \tag{4}$$

where λ is a scaling variable and the value of λ is 0.1 based on the preliminary experiments, $Tp(d)$ is document vector in topic-space and $Tm(d)$ is document vector in term space.

3.3 Query Sense Representation

We represent a query using a set of senses $S = \{s_1, s_2, s_3 ... s_n\}$ of the query which is generated using Wikiminer [23] word disambiguation. These raw senses are filtered and noise is removed by using tokenization, stemming and stop word removal techniques. Tokens generated from these senses are mostly bi-grams such as *jaguar car*, *sepecat jaguar*, *fender jaguar*, *mac os*. Other examples of senses are *panthera* and *south alabama jaguar football*.

3.4 Sense Similarity Measure

We define the similarity score between a document d_i and a sense s_j as a weighted sum of six different criteria:

$$SimSense(d_i, s_j) = \frac{|s_j|}{|d_i|} \sum_{k=1}^{6} w_k \cdot cmp_k(d_i, s_j) \tag{5}$$

The six criteria for *cmp* are exact sequence matching, semantic matching, partial matching in both term space and topic space of the document d_i for sense s_j. The exact sequence matching counts the number of occurrence of a sense s_j in document d_i. The semantic matching counts overlap of either exact or synonyms, and partial matching counts overlap of individual words in sense s_j and document d_i.

4 The Algorithm

We had developed a new algorithm called QSC that uses our new document representation and similarity measures. It includes three main steps: the first step is to group all the documents according to their similarity to the different senses of the user query; the second step is to iteratively optimize clusters by relocating documents from one cluster to another cluster based on the similarity between documents and the clusters; the third step is to rank the documents and clusters based on similarity with the user query.

4.1 Step 1: Initial Cluster Generation

The initial clusters are formed by calculating the similarity of each document with each user query sense and assigning each document to each cluster associated with the maximally similar sense. Each cluster is labeled with its associated sense. Documents that are not sufficiently similar to any sense are placed in a cluster labeled *general*. The set of initial clusters C consists of all the clusters that contains at least one document.

Algorithm 1 initClusters(query)

```
1: senses ← GetSenses(query)
2: documents ← GetDocuments(query)
3: C ← φ
4: for d ∈ documents do
5:     s ← φ
6:     for cs ∈ senses do
7:         s = find the maximum SimSense(d,cs)
8:     end for
9:     if s is Not Null then
10:        C_s ← GetCluster(C, s)
11:        if C_s is Null then
12:            C_s ← new Cluster(s)
13:            C ← C ∪ C_s
14:        end if
15:    end if
16: end for
17: return C
```

Algorithm 1 shows the pseudo code for generating the initial clusters. The function *initClusters* takes an argument *query* that is given by user on runtime. The function *GetSenses* provides all the senses of a query and the function *GetDocuments* provides all preprocessed documents along with their topics. The loop on Line 6 iterates over all senses and finds the maximum similar sense for document d. The function *GetCluster* is used to update an existing cluster that has cluster label s, or create a new cluster object C_s which is added to the list of clusters C.

4.2 Step 2: Cluster Optimization

Initial clusters were based on the similarity between documents and the senses. Base cluster labels can provide quality labeling of clusters. However the clusters, especially the general cluster may contain a mixed group of documents that might not be similar. We developed an iterative method to reassign some documents in order to improve cluster quality by increasing intra-cluster coherence and inter-cluster distinctiveness.

Algorithm 2 optimizeClusters(C)

```
1: relocate ← True
2: maxIter ← 200
3: while relocate is True ∧ maxIter > 0 do
4:     relocate ← False
5:     maxIter ← maxIter − 1
6:     for c_x ∈ C do
7:         UpdateMean(c_x)
8:     end for
9:     for c_x ∈ C do
10:        for d_x s.t. d_x ∈ set of documents of c_x do
11:            relocate ← RelocateDocument(c_x, d_x, C)
12:        end for
13:    end for
14: end while
15: C ← FilterEmptyClusters(C)
```

The pseudo code for this optimization is given in Algorithm 2. The algorithm repeatedly attempts to relocate documents to more appropriate clusters. The algorithm terminates if it fails to make any change, or if it reaches the predefined maximum number of iterations. The function $UpdateMean$ updates the average similarity scores of all clusters C by the using new similarity measure given in Equation 4. The function *RelocateDocument* removes document d_x from cluster c_x and adds it to another cluster $c_y \in C$ s.t. $c_y \neq c_x$ if the average similarity of the document d_x in cluster c_x is lower then the mean of c_y. The function $FilterEmptyClusters$ removes all clusters with no documents. One result of this algorithm is that closely related clusters can be merged by relocating all documents from one cluster to the other.

4.3 Step 3: Cluster Ranking

Users are interested in only those documents that are most closely related to the query. Therefore the ranking of clusters and documents are computed with respect to the query.

All the clusters were sorted, by calculating the relatedness score between the user query and the cluster label, using the term similarity measure $WikiSim$ [17].

The *WikiSim* is Wikipedia based similarity measure that computes relatedness between two terms. Documents in its own cluster were also sorted by calculating the similarity of a document to its mean in its own cluster. The ranked result list is then sent to user for browsing.

5 Results

The QSC was evaluated on two datasets, AMBIENT and MORESQUE.

- AMBIENT dataset consists of 44 ambiguous queries generated from Wikipedia [7]. For each of the 44 queries there are 100 snippets that are congregated from Yahoo. It also includes manually assigned class labels mapped to relevant snippets.
- MORESQUE dataset consists of 114 ambiguous queries and is an extension of AMBIENT dataset [24]. The AMBIENT dataset mainly has one word queries but MORESQUE contains more than one word queries.

5.1 Comparison 1

This work was compared with two most popular algorithms STC [35] and Lingo [26] and one recent algorithm SRCluster, [22] on a subset of AMBIENT dataset. This subset consist of 9 queries chosen by SRCluster [22]. The comparison is based on two evaluation criteria namely Entropy and Purity [39].These criteria are widely used in evaluating cluster quality and the coherence of a cluster. Entropy is defined as

$$Entropy = \sum_{i=1}^{k} \frac{n_x}{n} E(c_x); \;\; E(c_x) = -\frac{1}{\log k} \sum_{i=1}^{k} \frac{n_x^i}{n_x} \log \frac{n_x^i}{n_x} \tag{6}$$

Where n is total number of documents and n_x is total number of documents in cluster c_x. k represents number of classes in the dataset, n_x^i is the number of documents that belong to ith class. Purity is defined as

$$Purity = \sum_{x=1}^{k} \frac{n_x}{n} P(c_x); \;\; P(c_x) = \frac{1}{n_x} \max_i (n_x^i) \tag{7}$$

Where x is a cluster number, k is total number of clusters, n_x is total documents in cluster c_x, n is total number of documents in all clusters and i represents class that documents belongs to. Table 1 list the 9 queries of AMBIENT dataset chosen by the paper [22] on left column. The values of Entropy and Purity for STC, LINGO and SRCluster were also taken from the paper of SRCluster [22]. The values of QSC were computed on the same 9 queries to ensure fair comparison. The cluster quality is determined on the basis of lower entropy and higher purity of a given query. QSC outperformed other methods in almost all queries except the query *Zodiac*. The query *Zodiac* has entropy 0 and purity 1 for both SRCluster and QSC which indicates that both methods are perfect for this query.

Table 1: The Values of Entropy and Purity on a subset of AMBIENT Dataset

	Entropy				Purity			
Query	STC	LINGO	SRCluster	QSC	STC	LINGO	SRCluster	QSC
Aida	0.210	0.449	0.103	**0.018**	0.030	0.780	0.655	**0.878**
Camel	0.142	0.650	0.032	**0.019**	0.040	0.550	0.919	**0.928**
Indigo	0.390	1.000	0.071	**0.020**	0.000	0.400	0.867	**0.925**
Jaguar	0.190	0.495	0.041	**0.027**	0.290	0.290	0.888	**0.920**
Excalibur	0.880	1.000	0.080	**0.037**	0.000	0.090	0.774	**0.840**
Minotaur	0.150	0.400	0.068	**0.039**	0.220	0.300	0.760	**0.871**
Urania	0.229	0.700	0.299	**0.023**	0.140	0.348	0.782	**0.888**
Zenith	0.245	1.000	0.047	**0.017**	0.000	0.500	0.871	**0.958**
Zodiac	0.520	1.000	**0.000**	**0.000**	0.000	0.389	**1.000**	**1.000**

5.2 Comparison 2

The results on the larger dataset, which consist of all queries of AMBIENT and MORESQUE, based on purity and entropy were not given in [22]. However we found another recent paper [11] that compared nine algorithms using F1-measure on this large dataset. Therefore we compared our algorithm QSC with these nine algorithms using F1-measure calculated by taking the harmonic mean of precision and recall of the cluster [8]. The comparison was made between STC, LINGO, KeySRC [1], Curvature [12], SquaT++ [24, 11], B-MST [10], HyperLex [33], Chinese Whispers [2] and QSC. Figure 1 shows the percentage values of F1-

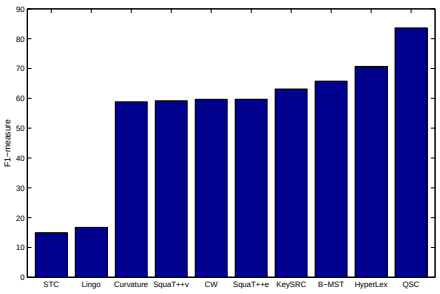

Fig. 1: Comparison of SRC methods

measure of 10 methods on combined dataset of AMBIENT and MORESQUE taken from the paper [11] and the computed value of QSC. Clearly the QSC performed significantly better than others and have the highest value 83.62 of F1-measure. Other evaluation criteria Adjusted Rand Index(ARI) and Jaccard Index(JI) are also used for comparing the clustering algorithms in paper [11].

However we believe that they are not suitable for these two datasets. More discussion is provided in the last part of section 5.4.

5.3 Comparison 3

The search results needs to be diverse and top ranked results should represent different senses of the user query. In order to determine the diversification of this work, the search results were evaluated based on **S-recall@K (Subtopic recall at rank K)** and **S-precision@r (Subtopic precision at recall)** [37] on combined dataset of AMBIENT and MORESQUE. The former evaluates the performance of the system based on K top-ranked results for number of topics of query q. S-precision@r measures the ratio of subtopics covered by minimum set of results at given recall r.

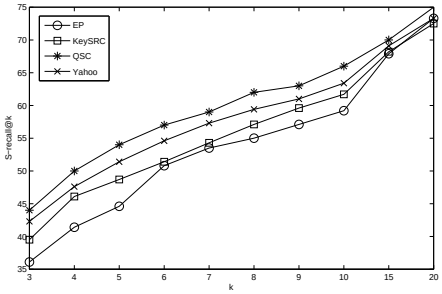

Fig. 2: S-recall@k on all queries

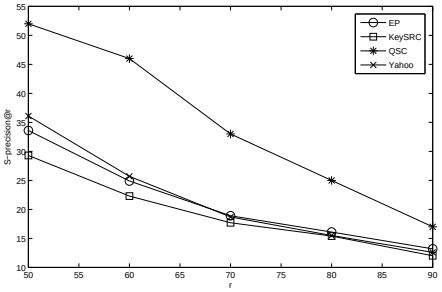

Fig. 3: S-precision@r on all queries

These two measures are used to compare search engines (Yahoo! and Essential Pages) that return ranked list of search results. The results returned by QSC were compared by flattening the clusters. The result list was formed by iterating through clusters and selecting top results. The clusters that only had one document were appended at the end to avoid noise.

Figure 2 and Figure 3 shows the S-recall@k and S-precision@r respectively for search results of Yahoo, Essential Pages(EP), KeySRC and QSC. The QSC performs relatively better in terms of S-recall@k and significantly outperformed others in terms of S-precision@r for the given values of k and r. This shows that QSC produced more diverse results than currently available search engines.

5.4 Further Analysis

The detailed analysis consists of three sub sections: the first discusses the cluster labels; the second discusses the processing time of the QSC, and the third discusses the cluster numbers and some observations about final clusters.

Table 2: Cluster labels of STC, LINGO and QSC of the query Jaguar

STC	LINGO	QSC
Jaguar Car	Auto Show	Jaguar Car
S-Type, Used Jaguar	Jaguar Parts	Jaguar E-Type
XK, 2006 2007, Price-Quotes and Reviews	Dealer Price Quotes and-Reviews	Jaguar XK
Ford Motor Company-Division	Ford Motor Company-Division	
Jaguar Cars		
Jaguar Panthera Onca	Jaguar Panthera Onca	Panthera
Jaguar Animal		
	Website of Fender Musical-Instrument	Fender Jaguar
Information	Jaguar Video	Mac OS X
New		SEPECAT Jaguar
		South Alabama Jaguar-Football

Cluster Label Analysis: The goal of the QSC algorithm is to generate a useful set of distinct clusters with informative labels.

Table 2 shows the cluster labels of the clusters generated for the query *Jaguar* by STC, LINGO and QSC (the cluster labels are not in ranked order). The labels for STC and LINGO were generated using the Carrot2 framework by adjusting the parameter of maximum clusters number to 8. Table 2 shows that the cluster labels generated by QSC provides more precise, intuitive and distinct labels than the cluster labels from STC and LINGO.

Processing Time: The QSC was evaluated on standalone workstation using Linux (64 bit) with Intel(R) Core(TM) i7-3770 CPU @ 3.40GHZ, 8GB RAM and 1TB HD. Figure 4 shows the processing time of all the queries in AMBIENT Dataset. The average time required for processing the query is under 1.0 second for both AMBIENT and MORESQUE datasets. Most of the queries were processed under one second with few exceptional cases. The maximum processing time was 6.3 seconds on a query *jaguar* because it had 54 senses to be processed. This processing time was reduced to 1 second by eliminating overlapping senses and processing only 10 distinct senses.

Strictly speaking, we cannot directly compare the processing time of other algorithms due to different machines and platforms. However we would like to give indications that word sense induction based algorithms (Curvature, Squat++, B-MST, HyperLex and Chinese Whispers) need to construct the graph to identify the senses from the huge corpus, whereas QSC extract the senses from the Wikipedia. Therefore the word sense induction based algorithms might require more processing time than QSC. The processing time of clustering, without

considering the time spent on graph construction, for all algorithms is under 1 second except for SquaT++ algorithm. The SquaT++v and SquaT++e spent around 28 and 21 seconds respectively for clustering results as described in their paper [11].

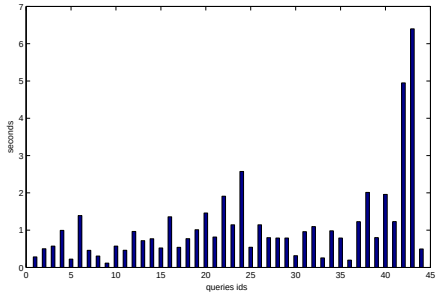

Fig. 4: Processing Time for All Queries

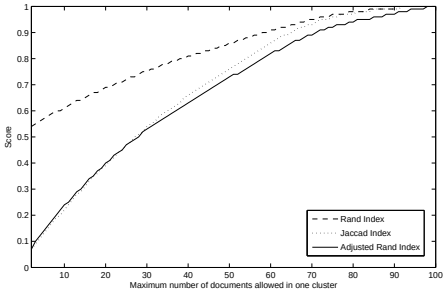

Fig. 5: RI, ARI and JI Analysis

Cluster Analysis: The average number of clusters for all queries in the AMBIENT dataset was 7.84 i.e on average 7-8 clusters are formed for each query. The average number of clusters for all queries in the AMBIENT and MORESQUE datasets was 5.4. There were a few queries with a high number of clusters and the maximum number of clusters was 18 for the query *Monte Carlo*. In contrast the query *Life on Mars* just had 1 big cluster. The reason for having many clusters was the large number of distinct query senses. The query *Life on Mars* had very few senses and they were overlapping with each other, e.g. *Life on Mars (TV series), Life on Mars (U.S TV series)*, that causes single cluster for the query.

The QSC provided a more fine-grained clustering solution than the gold standard (manually labeled search results). The gold standard for the query *jaguar* had 7 clusters but QSC solution provided 9 clusters. The three clusters *jaguar car, jaguar e-type and jaguar xk* in QSC were sub clusters of gold standard *jaguar car*.

The QSC was not compared with other algorithms using index based evaluation measures (ARI and JI) because these measures have many issues [30, 21]. One of the problems is that they do not handle fine-grained clustering solutions. If a gold standard G has a cluster g_i that contains 90 documents and clustering C has clusters c_j, c_{j+1} and c_{j+2} that contain all 90 documents then ARI and JI will penalize the clustering solution heavily. However the fine-grain clustering solution is consistent with the coarser solution and should not be penalized heavily. In fact it may even be better solution because it provides the distinctiveness that are not provided by the gold standard. ARI and JI do not measure this.

Figure 5 shows the phenomena of heavy penalty of ARI and JI as compared to Rand Index (RI) [16] on sub clusters. This experiment was performed on the AMBIENT and MORESQUE datasets by evaluating the perfect sub-clusters

that gradually increased the limit of the maximum number of documents allowed in a cluster from 2 to 98. All the documents were perfectly assigned to the clusters and the values of RI, ARI and JI were computed at each iteration. The lowest value of RI, ARI and JI were 0.54, 0.05 and 0.04 respectively when the maximum allowed number of documents in sub clusters were 2. Figure 5 shows ARI and JI penalize small clusters and small sub-clusters heavily. The gold standard in our dataset had very unbalanced number of clusters. A few clusters were very small, and had 2 documents in a cluster and other were very large and had more than 90 documents. It was observed that the comparison based on ARI and JI is suitable only when the gold standard do not have sub clusters and all the clusters have almost the same number of members.

6 Conclusion

This paper presents a new description-centric search result clustering algorithm QSC that exploits query senses to generate meaningful cluster labels and use syntactic and semantic features of documents to generate quality clusters.

This paper shows that QSC outperforms existing algorithms. QSC is computationally inexpensive and provides better quality clusters with meaningful labels as compared to other algorithms, hence it has the potential to be applied to real time search result clustering applications.

The future direction for this work is to use Google WebIT and ukWac corpus along with Wikipedia to enhance the quality of query senses. The similarity measure and documentation representation are the key factors and a better similarity measure could bring more improvement. The greedy search in step 2 of the QSC could be improved to avoid local optima, by using the query senses in addition to document similarity. The currently used topic detection technique is not as good as state-of-the-art topic detection techniques such as LDA [3]; using LDA to detect topics from search results by considering query senses may further improve this work.

References

1. A. Bernardini, C. Carpineto, and M. D'Amico. Full-subtopic retrieval with keyphrase-based search results clustering. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 206–213. IET, 2009.
2. C. Biemann. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80. Association for Computational Linguistics, 2006.
3. D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
4. S. Branson and A. Greenberg. Clustering web search results using suffix tree methods. Technical report, Stanford University, Tech. Rep. CS276A Final Project, 2002.

5. C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17, 2009.
6. C. Carpineto and G. Romano. Exploiting the potential of concept lattices for information retrieval with credo. *Journal of universal computer science*, 10(8):985–1013, 2004.
7. C. Carpineto and G. Romano. Ambient dataset, 2008.
8. D. Crabtree, X. Gao, and P. Andreae. Improving web clustering by cluster selection. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 172–178. IEEE, 2005.
9. D. Crabtree, X. Gao, and P. Andreae. Query directed clustering. *Knowledge and Information Systems*, pages 1–37, 2012.
10. A. Di Marco and R. Navigli. Clustering web search results with maximum spanning trees. In *AI* IA 2011: Artificial Intelligence Around Man and Beyond*, pages 201–212. Springer, 2011.
11. A. Di Marco and R. Navigli. Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics*, (Just Accepted):1–76, 2013.
12. B. Dorow, D. Widdows, K. Ling, J.-P. Eckmann, D. Sergi, and E. Moses. Using curvature and markov clustering in graphs for lexical acquisition and word sense discrimination. *arXiv preprint cond-mat/0403693*, 2004.
13. P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 801–810. ACM, 2005.
14. F. Giannotti, M. Nanni, D. Pedreschi, and F. Samaritani. Webcat: Automatic categorization of web search results. *SEBD03*, pages 507–518, 2003.
15. M. Hearst and J. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84. ACM, 1996.
16. L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.
17. S. Jabeen, X. Gao, and P. Andreae. Harnessing wikipedia semantics for computing contextual relatedness. In *PRICAI 2012: Trends in Artificial Intelligence*, pages 861–865. Springer, 2012.
18. K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proceedings of the 13th international conference on World Wide Web*, pages 658–665. ACM, 2004.
19. Y. S. Maarek, R. Fagin, I. Z. Ben-Shaul, and D. Pelleg. Ephemeral document clustering for web applications. 2000.
20. I. Masłowska. Phrase-based hierarchical clustering of web search results. In *Advances in Information Retrieval*, pages 555–562. Springer, 2003.
21. M. Meilă. Comparing clusteringsan information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007.
22. Y. Meiyappan, N. C. S. N. Iyengar, A. Kannan, Y. D. Suyoto, T. Suselo, T. Prasetyaningrum, R. Tlili, Y. Slimani, S. Dufreche, M. Zappi, et al. Srcluster: Web clustering engine based on wikipedia. *International Journal of Advanced Science and Technology*, 39(1):1–18, 2012.
23. D. Milne and I. H. Witten. An open-source toolkit for mining wikipedia. *Artificial Intelligence*, 2012.

24. R. Navigli and G. Crisafulli. Inducing word senses to improve web search result clustering. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 116–126. Association for Computational Linguistics, 2010.
25. C. L. Ngo and H. S. Nguyen. A tolerance rough set approach to clustering web search results. In *Knowledge Discovery in Databases: PKDD 2004*, pages 515–517. Springer, 2004.
26. S. Osiriski, J. Stefanowski, and D. Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent information processing and web mining: proceedings of the International IIS: IIPWM04 Conference held in Zakopane, Poland*, page 359, 2004.
27. T. Pang-Ning, M. Steinbach, and V. Kumar. Introduction to data mining. *WP Co*, 2006.
28. J. Park, X. Gao, and P. Andreae. Query directed web page clustering using suffix tree and wikipedia links. In *Advanced Data Mining and Applications*, pages 91–99. Springer, 2012.
29. P. Pirolli, P. Schank, M. Hearst, and C. Diehl. Scatter/gather browsing communicates the topic structure of a very large text collection. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–220. ACM, 1996.
30. A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, volume 410, page 420, 2007.
31. G. Salton and M. J. McGill. Introduction to modern information retrieval. 1986.
32. B. Stein and S. M. Zu Eissen. Topic identification: Framework and application. In *Proc. International Conference on Knowledge Management*, volume 400, pages 522–531, 2004.
33. J. Véronis. Hyperlex: lexical cartography for information retrieval. *Computer Speech & Language*, 18(3):223–252, 2004.
34. P. Worawitphinyo, X. Gao, and S. Jabeen. Improving suffix tree clustering with new ranking and similarity measures. *Advanced Data Mining and Applications*, pages 55–68, 2011.
35. O. Zamir and O. Etzioni. Web document clustering: a feasibility demonstration. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 46–54, New York, NY, USA, 1998. ACM.
36. O. Zamir, O. Etzioni, O. Madani, and R. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 287–290. MIT Press, 1997.
37. C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 10–17. ACM, 2003.
38. X. Zhang, X. Hu, and X. Zhou. A comparative evaluation of different link types on enhancing document clustering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 555–562. ACM, 2008.
39. Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. *Machine Learning*, 2001.