Query-context Search Result Clustering basing on Graphs*

Michał Meina

Faculty of Mathematics and Computer Science
Nicolaus Copernicus University
Chopina 12/18, 87-100 Toruń

Abstract. This papers deals with search result clustering by introducing a novel approach to a in-context clustering. Basing on additional information provided by the user in query we can develop custom document grouping. This work is a preliminary (extended abstract) version.

1 Introduction

Information Retrieval Systems (e.g Carrot2 Clustering Engine[1] or Grouper [17]) often provides mechanisms to organize retrieved results in clusters in order to simplify process of navigating to an interesting document. In those cases search results clustering is a type of postprocessing result set. According to [8] giving a user simple mechnism to correct or change clustering can significantly improve Information Retrieval (IR) task. Although search-result clustering is currently the subject of extensive research there is lack of approches that refolmulate interaction scenario for clustering engines. We state that this ability can be crucial in narrowed dataset where differences beetwen clusters could be very subtle. This paper introduces query-context search result clustering method based on graphs.

Our system operates on result search taking an additional query(ies) q_{desc} and makes query-context document clustering. Instead of treating document-to-cluster belonging as a document property we will allow same documents to be assigned to entirely different clusters depending it to additional queries (clusering intention). Our alghoritm generates a cluster description first and then assigns documents into one or more clusters matching document with description. This technique is called description-comes-first strategy [14]. Descriptions are generated in a way that takes into consideration user query which can lead to a situation where the same documents set generates different clustering. Search task introduced by our method is ilustrated by examaple on Table 1.

search query	additional queries	intended clusters
	iphone	approximation of standard clustering
iphone	apple	product announcements official statements software announcements
	microsoft	product comparision official statements about counterpart products

Table 1.

For document representation we developed graph model on top of classic Vector Space Model (VSM). Idea that stands behind using graphs comes from efficiency-effectivity tradeoff. In online query processing we cannot do much computation, so we use "richer" model which is constructed offline and then we apply

* This work was supported by grant N N516 077837 from the Ministry of Science and Higher Education of the Republic of Poland.

[1] http://search.carrot2.org/

CONCURRENCY, SPECIFICATION AND PROGRAMMING
M. Szczuka et al. (eds.): Proceedings of the international workshop CS&P 2011
September 28-30, Pułtusk, Poland, pp. 346-352

“faster” alghoritm in query processing phase. As it was pointed in [7] semantic anylsis such a Latent Semantic Indexing (LSI) make extension to VSM that captures information about co-occurences of higher rank. In models based on graphs those co-occurences are expressed directly in the model, which facilitates the use of this information in the clustering task.

Our clustering scenario does not focus on optimising standard quality criterion of clusters but according to [10] high score on internal measure do not necessarily result in effective information retrieval applications. Our goal is to create strong boundaries beetwen clusters, and scatter topics across individual groups to be much “orthogonal” taking q_{desc} as criterion of orthogonality. Still we can maintain acceptable high quality of clusters which are organized in context of a query.

Contributions of this paper are as follows: (1) introduction of Query-Summarize Graph - a graph structure which can be used in clustering search results in context of a user query (2) definition of similarity measurements between document and term graph.

2 Related Work

Earliest attempts to cluster-based search engines depend on clustering whole documents collection and then adding this information into results. For example in [1] interactive dendrogram was presented alongside keyword search result.

Leouski et al. [8] indicates that clustering of search results gives better user-feedback and shortends time to retrieve documets. Morover this technique gives an opportunity to a user to discover some additional knowledge about dataset. Conclusions of this work tells that giving some means to a user in order to correct or completely change classification structure (e.g. by seting cluster boundaries) can significantly improve searching.

In [6] authors introduce *scatter/gather* approach to present and navigate within documents. This mechanism redefine search task providing “query without typing” user interface strongly . Search process starts with presenting large clusters of whole document collection and then it allows user to select (*gather*) interesting clusters or items. The narrowed set is merged into one collection and clustered again (*scattered*). This step is repeated as long as the whole dataset is narrowed sufficiently. Mentioned paper is a good example of IR system that strongly faciliate clustering.

There is a number of papers that deal with web-search result clustering (e.g. [16] or [5]). Clustering of webpages slightly differ from regular document clustering. The problem here is in refining terms extrected from webpages since words can differ from its context or developing documents features that takes into account web environment. Gelgi et al. [4] introduces graph based approach in refining terms from a web-search result. They built a undirectet graph of terms in which edges weight corresponds to co-occurence in search results. Their varation of PageRank alghoritm called *TermRank* was able to distinguish discriminative, ambigious and common words in order to refine terms in documents before clustering. *TermRank* was significantly better than classic tf-idf weightening scheme before clustering. foucesd on

There is a number of possibilites of how we can model documents as graphs. Starting with simple ones presented in [12], where authors describes six methods how we can map terms extracted from documents as graphs. Our model is very similar to *simple* model from this paper, but we are more foucesd on investigation different techniques of graph construction. In [9] authors present two alghoritms that make keyword-summarization of documents. The experimental results reports gaining high f-masure of keyword extraction against summarization prepared by humans.

More sophisticated graph models are variation of Conceptual Graphs (CG’s) [13] which provides formal definiton of semantic representation. Automatic construction of CG’s is a difficult problem and not necesseraly easy aplicable in Information Retrieval taks. In [11] authors was using simplified CG’s that provided more semantic-oriented analysis. They slightly improved retrieval performance comparing to standard vector space model by defining documents similarity to take into consideraton graph structure. In [3] authors were able to capture domain knowledge in graph represenation, stating that this is natural and very intuitively method of representing knowledge.

Finally [15] Multi-Level Association Graphs (MLAG's) was able to subsume various IR models into one common also defining similarity mesaure beetwen on-line query by Random Walk on precomputed MLAG. None of mentioned works consider construction of graph in a query processing pipeline. Moreover, none of these works did not investigate different technique of the model instantiating.

3 Graph generation

Graph generation is divided into two phases: (1) first we construct graph depending on search result and user query (2) we decompose it into number of disjoint subgraphs.

Query-Summarize Graph is a weighted, undirected graph $G = (V, E)$ whose vertices corresponds to terms ($t \in T$) in documents and edges weights are assigned according to semantic distance between terms (in our case induced from results). There is a set of distinguished nodes whose corresponding terms occurred in a query $q = (t_1, t_2, ...)$. Graph is constructed iteratively - nodes and edges are being added to graph basing on previously added nodes until stop-condition is satisfied.

Graph generation starts by adding nodes for terms extracted from query whereby edges between them are immediately created making small local clique at begining. In $n + 1$ step of alghoritm we add term that are semantically closest to previously added terms in step n.

Adding nodes to graph is constrained by k semantically closest terms threeshold for each visited node. Naive implementation could depend on choosing this parameter as fixed or depending it on fixed semantic distance thresshold. Through transitivity of similarity relation the edges would be created only between nodes added in n and $n + 1$ step that in the end can result in graphs that will have unsatisfactionary low clustering coeficient. Moreover stop-condition could be enforced only by choosing unintuitive depth parameter that stops algorithm after specified number of steps.

Our alghoritm changes the number of terms to be added basing on the number of terms added in previuos step. We follow simple strategy: if we have added previously a large number of nodes (which stand for possibly exploration of a new topic) we should add a small number of terms which are semantically very close to this one (which contribute to in-depth description of the topic). In first step we add a $InitialSeed$ number of semantically closest terms and in every next step we add k closest ones where

$$k = \frac{InitialSeed}{|NodesAddedInPreviousStep|}$$

Require: $depth, q$
  $G(V, E) \leftarrow Graph()$
  $ToVisit \leftarrow List(q), Visited \leftarrow Set()$
  for all $(t_i \in q, t_j \in q) \wedge i \neq j$ **do**
    $Add\, t_i\, and\, t_j\, into\, V$
    $Add\, (t_i, t_j, sim(t_i, t_j))\, into\, E$
  end for
  while $depth \neq 0 \vee ToVisit = \emptyset$ **do**
    for $t \in ToVisit$ **do**
      $k \leftarrow ChangeTopThreeshold()$
      for $t_c \in TopSimTerms_k(t, T)$ **do**
        if $t_c \notin Visited$ **then**
          $Add\, t_c\, into\, ToVisit$
        end if
        $Add\, t_c\, into\, V$
        $Add\, (t, t_c, sim(t, t_c))\, into\, E$
      end for
      $Add\, t\, to\, Visited$
    end for
    $depth \leftarrow depth - 1$
  end while
  return G

Later we show that in our experiments this method tends to stop after realtively small number of steps outputing graph with gratifying clustering coefficient[2].

After constructing graph we decompose it to disjont subgraphs by discarding edges that connects modules (generalization of connected componets) in graph. We use louvain method [2] to find those modules which

[2] Our approach of tunnning k parameter in this preliminary work is very simple. Nevertheless this step makes great space for research in final version, since according to [] semantic similarity distribution and similar term overlaping is stronly based on topics captured in documents

is an optimisiation of modularity maximization alghoritm. This alghoritm tries to partition the graph into modules that have strong internal connection and sparese connectivity beetwen modules. Modularity is a benefit function that in this case measures quality of particular division according to information how module could be connectd at random

In search task our system is using those subgraph as structural cluster descriptions.

4 Query-Summarize Graphs structural analysis

For the needs of our experiments we crawled web for newswire text from four different sources. For data prepearation we apply small hand-maded stop words list and stemming. Since articles from newswires have large number of named entieties we developed heurestic that identifies and normaliazes those basing on n-gram model. As semantic similarity we have used cosine distance on tfidf model

$$sim(t_i, t_j) = \frac{\boldsymbol{V(t_i)} \cdot \boldsymbol{V(t_j)}}{|\boldsymbol{V(t_i)}||\boldsymbol{V(t_j)}|},$$

where $\boldsymbol{V(t_i)}$ is term column in tf-idf weighted document-term matrix.

Search results were divided into two groups using keyword search and time span search. Graphs was genereted by set of human-choosed queries q_i using first 100, 200, 500, 1000 documents from search results. On figures and tabels $q_i - ts$ and $q_i - q_j$ stands for graph generation by query q_i respectively on time span search result and keyword search results by query q_j.

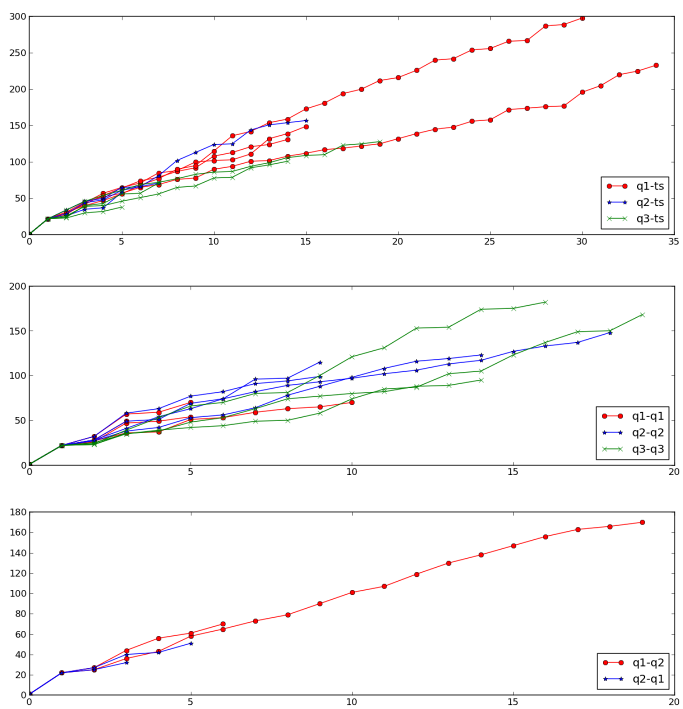

Fig. 1. Nodes count (y-axis) and number of steps of alghoritm (x-axis)

For large set of randomly choosed queries graph generation stops in not more than fifty steps. Most of the queries picked randomly were having large tf and small idf value and those terms generates graphs where node count was around $InitialSeed$ number. That indicates when we are quering with terms that make small contribution in any topic we end with undescriptive graphs. On the other hand quering with highly descriptive words (e.g. named entieties) generates graphs with large number of topics. Information about number of nodes in Q-S Graph on every step of alghoritm shows steady growth with few rapid increases. This follows our intuition in tunning k parameter - nodes added when k parameter was low tends to make strong bounds with parts of graph which in the future become modules than with other parts of graph.

Another promising result is that graph size tends to have higher correlation with query that with the size of result set.

After generating q-s graphs we need to decompose it to disjont subgraphs. Information about modularity of Q-S Graphs we measure by Average Clustering Coefficient. This measure is a property of a network that indicates how nodes in graphs tends to cluster together. Assuming that $N_v \subset V$ is set of immediate neighours of node v. Clustering Coefficient is defined as follows:

$$C = \frac{1}{n} \sum_{v \in V} C_v, \; where$$

$$C_v = \frac{2|\{e_{jk}\}|}{|N_v|(|N_v| - 1)} : v_j, v_k \in N_v, e_{jk} \in E$$

DocNo	q1-ts	q2-ts	q3-ts	q1-q1	random
100	0.5155	0.8882	0.6303	0.6395	0.1759
200	0.5161	0.6241	0.6868	0.5741	0.0987
500	0.6712	0.4949	0.8483	0.4725	0.0675
1000	0.5172	0.4378	0.7991	0.7230	0.0797

Table 2. Average Clustering Coefficient for Q-S Graphs and Alberti Barabasi random graph model

We see that Average Clustering coefficient was significantly larger than random graph of similar size. That indicates that Q-S Graph generated with tunning strategy ofg k parameter tends to have nodes arranged in highly coupled groups.

Most interesting point is how cluster descriptions look like if we query for semantically close terms. In order to evaluate clustering features in this preliminary work we take case-driver evaluation choosing another three queries q_a, q_b, q_c. But in this case q_a and q_b was synonyms and q_c was generalization of two others and we generated graphs against time span search results. As we expected Q-S Graphs for queries that were synonymys was having similar size (repsectivley 143 nodes and 279 edges to 158 nodes and 298 edges). Graph from last query was significanlty larger (176 nodes and 402 edges), capturing more terms. Number of clusters was respecively 14, 19, 17. It was not expected that number of clusters will be different for synonyms.

Terms for every cluster description was subjected to a manual evaluation and results were encouraging. For synonyms the largest modules were almost identicall. More than 30% of cluster descriptions were having large number of common words and the differences comes from different context in which each term tends to be used in newswire texts. Other cluster descriptions and their difference was very hard to interpret without examining documents. Largests clusters in graph for query q_c was very similar to largest clusters for g_a and q_b but was enriched by new words. New words was covering larger semantic field of generalized query.

Our preliminary experimental results shows that our alghoritm stops outputing graph with high clustering tendency. That allows us to decompose it into number of disjont subgraphs $G'_1, G'_2, ..., G'_k$, which we later treat as cluster-descriptions. After exmaining words in decomposed graphs we encountered that distribution of their features follow our intuition in query-context search.

5 Application in Information Retrieval System

Q-S Graphs can be employed easily in search result clustering based on user interaction. After retrieving results documents could be grupped according to an initial or additional query (which can be applied repetedly). We propose score function that will classify every search result according to cluster descriptions. We assume that document is represented as a vector of extracted terms $d = [t_1, ..., t_n] : t_i \in T$, cluster description is subgraph of G that $G' = (E' \subset E, V' \subset V)$, with weight function w.

First we need to define measure of corespondece of term x to graph G' in context of a term t:

$$P_{G',t}(x) = \sum_{y \in N(x)} \frac{sim(t,y)w(x,y)P_{G',t}(y)}{|N(x)|}$$

At this point we can define two score measures that indicates document to cluster belonging. Strong score (which filters documents whose nodes are presented in graph G')

$$\widehat{score}_{G'}(d) = \sum_{t \in d \wedge t \in V'} P_{G',t}(t)$$

Strong score can significantly narrow document collection depending it on the size of graph G'. In order to classify every document we can defin smooth score, as:

$$score_{G'}(d) = \sum_{t \in F(d)} P_{G',t}(t)$$

$$F(d) = \{t' : sim(t', y) = max(\{sim(t, y) : y \in V'\})\}$$

One drawback of constructing Q-S graphs in on-line query procesing is that we need to construct tf-idf weighted document-term matrix for retrieved document set and make $|V||T|$ computation of similarity distance. Beside simple optimisiation that we can use in sparse vectors computation we can redefine semantic distance function sim as for example number of co-occurences two terms in documents like in [4] $sim(t_i, t_j) = |\{d \in D : t_i \in d \wedge t_j \in d\}|$.. Other possibility is using it as precomputed from whole or fraction of dataset or even taken from elsewhere in the first place. This can lead us to application in another IR technique called Query Expansion[3]. After construction Q-S graph for query $q = (t_1, t_2, ..)$ we can discover (e.g *TermRank*) from it additional terms t'_1, t'_2... which can contribute to fast inverted index scan (keyword search) by query $q' = (t_1, t_2, .., t'1, t'_2)$, retrieving more potentialy interested documents.

References

1. Robert B. Allen, Pascal Obry, and Michael Littman. An interface for navigating clustered document sets returned by queries. In *Proceedings of the conference on Organizational computing systems*, COCS '93, pages 166–171, New York, NY, USA, 1993. ACM.
2. Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.
3. Madalina Croitoru, Bo Hu, Srinandan Dasmahapatra, Paul H. Lewis, David Dupplaw, Alex Gibb, Margarida Julià-Sapé, Javier Vicente, Carlos Sáez, Juan Miguel García-Gómez, Roman Roset, Francesc Estanyol, Xavier Rafael Palou, and Mariola Mier. Conceptual graphs based information retrieval in healthagents. In *CBMS*, pages 618–623. IEEE Computer Society, 2007.
4. Fatih Gelgi, Hasan Davulcu, and Srinivas Vadrevu. Term ranking for clustering web search results. In *WebDB*, 2007.
5. Gaojie He, Co supervisor Robert Neumayer, Gaojie He, Robert Neumayer, and Kjetil Norvag. Learning to cluster web search results. In *In Proc. of SIGIR '04*, pages 210–217, 2004.
6. Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. pages 76–84, 1996.
7. April Kontostathis, William M. Pottenger, and Ph. D. Detecting patterns in the lsi term-term matrix. In *In Proceedings ICDM'02 Workshop on Foundations of Data Mining and Discovery*, 2002.
8. Anton V. Leouski and W. Bruce Croft. An evaluation of techniques for clustering search results. Technical report, 1996.

[3] the process of reformulating a seed query to improve retrieval performance

9. Marina Litvak and Mark Last. Graph-based keyword extraction for single-document summarization. In *Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization*, MMIES '08, pages 17–24, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.
10. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, July 2008.
11. Sonia Ordoñez Salinas and Alexander Gelbukh. Information retrieval with a simplified conceptual graph-like representation. In *Proceedings of the 9th Mexican international conference on Advances in artificial intelligence: Part I*, MICAI'10, pages 92–104, Berlin, Heidelberg, 2010. Springer-Verlag.
12. Adam Schenker, Horst Bunke, Mark Last, and Abraham Kandel. Graph-theoretic techniques for web content mining. 2005.
13. John F. Sowa. Conceptual graphs. In *Information Processing in Mind and Machine*, pages 39–44. Addison-Wesley, 1984.
14. Jerzy Stefanowski and Dawid Weiss. Extending k-means with the description comes first approach. *Control and Cybernetics*, (4), 2007.
15. H. F. Witschel. Multi-level association graphs - a new graph-based model for information retrieval. In *Proceedings of the HLT-NAACL-07 Workshop on Textgraphs-07, New York*, New York, USA, 2007.
16. Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. pages 46–54, 1998.
17. Oren Zamir and Oren Etzioni. Grouper: a dynamic clustering interface to web search results. In *Proceedings of the eighth international conference on World Wide Web*, WWW '99, pages 1361–1374, New York, NY, USA, 1999. Elsevier North-Holland, Inc.