Carrot^2 and Language Properties in Web Search Results Clustering

Jerzy Stefanowski and Dawid Weiss

Institute of Computing Science, Poznań University of Technology,
ul. Piotrowo 3A, 60–965 Poznań, Poland,
Jerzy.Stefanowski@cs.put.poznan.pl,
Dawid.Weiss@cs.put.poznan.pl

Abstract. This paper relates to a technique of improving results visualization in Web search engines known as *search results clustering*. We introduce an open extensible research system for examination and development of search results clustering algorithms – Carrot^2. We also discuss attempts to measuring quality of discovered clusters and demonstrate results of our experiments with quality assessment when inflectionally rich language (Polish) is clustered using a representative algorithm - Suffix Tree Clustering.

Keywords: Information Retrieval, Web Browsing and Exploration, Web Search Clustering, Suffix Tree Clustering

1 Introduction

The Internet, not even two decades old, has already changed the world by wiping physical distance limitations, bringing down information access restrictions and causing unprecedented growth of new technologies and services. However, with new possibilities come new boundaries. In particular, as it quickly turned out, easier access to large volumes of data was accompanied by an increasing level of difficulty in its organizing, browsing and understanding. Search engines thus quickly gained popularity and became the most popular tool for locating relevant information on the Web.

The user interface of the most popular modern search engines is based on keyword-based queries and endless lists of matching documents (all of the most popular search engines listed by Nielsen//NetRankings[1] use it). Unfortunately, even when exceptional ranking algorithms are used, relevance sorting inevitably promotes quality based on some notion of popularity of what can be found on the Web. If an overview of the topic, or an in-depth analysis of a certain subject is required, search engines usually fail in delivering such information. It seems natural to expect that search engines should not only return the most popular documents matching a query but also provide another, potentially more

[1] (01/2002) http://www.searchenginewatch.com/reports/netratings.html

comprehensive, overview of the subject it covers. Certain inventions have already been introduced in the commercial world: query expansion suggestions in Teoma or Infonetware, graphical visualization of results in Kartoo, image-based interface in Ditto or Google Images.

One of the most promising approaches, both commercially and research-wise, is to automatically group search results into thematic categories, called *clusters*. Assuming clusters descriptions are informative about the documents they contain, the user spends much less time following irrelevant links. He also gets an excellent overview of subjects present in the search result and thus knows when the query needs to be reformulated. Because the clustering process is performed dynamically for each query, the discovered set of groups is apt to depict the real structure of results, not some predefined categories (to differentiate this process from off-line clustering or classification, we call it *ephemeral* [1], or *on-line clustering* [2]). Our research follows this direction also taking into account certain characteristic properties of the Polish language – rich inflection (words have different suffixes depending on their role in a sentence) and less strict word order in a sentence (compared to English).

In this paper we introduce an open, flexible and free research project for building search results clustering systems – Carrot^2. We hope this framework could facilitate cross-comparison of algorithms and spawn new ideas for visualization of search results. We also discuss the topic of measuring the quality of on-line search results clustering, and demonstrate certain results acquired from application of an entropy-based measure to clustering Polish queries. We tried to analyze the influence of several term conflation algorithms on the quality of clusters acquired from a representative algorithm – *Suffix Tree Clustering*, in order to show differences between clustering two inflectionally different languages (Polish and English). This is an extension of our preliminary experiments presented in [3] – we enlarged the set of queries and included English queries as well as Polish.

2 Related Works

Clustering algorithms have been present in Information Retrieval for a long time, a comprehensive review of classic methods can be found in [4]. One of their common applications was to organize large volumes of semi numerical data into entities of higher abstraction level, easier to perceive by humans. *Agglomerative Hierarchical Clustering* (AHC) and *K-means* algorithms gained widespread popularity when speed was of no such critical importance, because processing was performed off-line and only once in a while. These algorithms, used successfully in many fields were quickly transformed to the domain of search results clustering. However, their computational complexity, difficult tuning of parameters and sensitivity to malicious input data soon raised the need of improvements.

The first proposal for *on-line and dynamic* search results clustering was perhaps presented in the Scatter-Gather system [5], where non-hierarchical, partitioning Fractionation algorithm was used. Undesired and troublesome high

dimensionality of term frequency vectors was addressed in [6], where two derivations of graph-partitioning were presented. Simple terms were replaced with "lexical affinities" (pairs of words commonly appearing together in the text) in [1], with a modification of AHC as the clustering algorithm. A different approach to finding similarity measure between documents was introduced in Grouper [2] and MSEEC [7] systems. Both of these discover *phrases* shared by document references in the search results and perform clustering according to this information. The former system introduces novel, very efficient *Suffix Tree Clustering* algorithm (described in Sect. 3), while the latter proposes utilization of LZW compression algorithm to find recurring sequences of terms in the input. STC produces flat, but overlapping clusters, which is usually perceived as an advantage, because documents tend to belong to more than one subject. An extension of STC producing hierarchical structure of clusters was recently proposed in [8].

All of the already mentioned algorithms were designed primarily to work on English texts and take English grammar and inflection into account. According to our knowledge, the only search results clustering technique working on sequences of characters, as opposed to words, was presented in [9].

3 Suffix Tree Clustering Algorithm

In this section, we present a very brief description of the Suffix Tree Clustering algorithm (STC), which we decided to perform our experiment on. An uninitiated Reader may want to refer to [10] for an in-depth overview.

Suffix Tree Clustering algorithm (STC) works under assumption that common topics are usually expressed using identical phrases (ordered sequences of words). For example, if two documents contain a phrase "Ben and Jerry's ice cream", then we may assume these documents are somehow relevant to each other and could be placed in one group. The same process applies to all other phrases shared by at least two documents. In the above example, the ice cream flavors (such as "chunky monkey" or "cherry garcia") would also become candidates for separate groups. One can notice that the discovered clusters could form a hierarchical structure. The original STC, however, produced the results as a set of (possibly overlapping) clusters, without any hierarchy on top of them. Several methods based on STC were successively proposed for this particular area [9, 8].

The main advantage of STC over methods utilizing term frequency distribution only (*keywords*) is that phrases are usually more informative than unorganized set of keywords, and can be directly used to label the discovered clusters, which in other clustering algorithms becomes a problem in itself. STC is organized into two phases: discovering sets of documents sharing the same phrase (*base clusters*) and combining them into larger entities (*final clusters*).

Base clusters are discovered by creating a generalized suffix tree for all terms in the search result. The description of this process is fairly complex (we encourage the Reader to refer to [11, 10] for details), so we should omit its description here. In the end, the result of this step is a set of phrases shared by at least

two documents, with links to all documents they occur in. This part of STC is characterized by (at least theoretically) linear complexity and can be performed incrementally.

Each base cluster a is described by its associated phrase m_a and the set of documents that phrase occurs in d_a. We then calculate a *base cluster score* defined as: $s(a) = |m_a| \times f(|m_a|) \times \sum_{w \in m_a}(tfidf(w))$, where $|m_a|$ is the number of terms in phrase m_a, $f(|m_a|)$ is a function penalizing short-phrases (because we prefer longer, more descriptive sequences of terms), and $\sum_{w \in m_a}(tfidf(w))$ is a sum of standard Salton's *term frequency-inverse document frequency* term ranking factor for all terms in phrase m_a.

Only base clusters with the score higher than a *minimal base cluster score threshold* are promoted for the second step of the algorithm – merging. Merging of two base clusters a and b is performed, when their document sets d_a and d_b overlap more than a predefined *merge threshold*. STC thus way groups documents which relate to a subject expressed by more than one phrase. For example, base clusters "Ben and Jerry's ice cream" and "chocolate fudge" would hopefully express documents relevant to this particular flavor of ice creams.

The merge process is in fact a variation of agglomerative hierarchical clustering algorithm: if α denotes the *merge threshold* and $|x|$ denotes the number of documents in cluster x, the binary merge criterion for the AHC algorithm is defined by: $similarity(a,b) = 1 \Leftrightarrow \left(\frac{|d_a \cap d_b|}{|d_a|} > \alpha\right) \wedge \left(\frac{|d_a \cap d_b|}{|d_b|} > \alpha\right)$.

Eventually, the base cluster score is recalculated for merged clusters and the top-scoring clusters are displayed to the user.

4 Carrot² Search Results Clustering Framework

We started asking questions about feasibility of search results clustering in Polish around the time Grouper was available on-line. Unfortunately, we could not proceed with further experiments because that system was never released in source code. We created Carrot then – an open source implementation of STC – where we introduced certain features dedicated to processing of the Polish language: quasi-stemming, simple lemmatization method and a large corpora of Polish texts, from which stop words were extracted. These were our own contribution because morphological analysis of Polish requires significant effort and hence, understandably, out of several existing algorithms, none is available free of charge. Carrot was used to perform initial experiments and user studies, also bringing new questions regarding sensitiveness of STC's thresholds, influence of stemming quality and inflection. Although Carrot was a well designed prototype, its development was very rapid and several design decisions limited further experiments. Having learned from that example, we decided to create a new, open and extensible research framework for experimenting with search results clustering - Carrot^2.

We have observed, that almost every search results clustering system described in literature could be split into reusable and fairly autonomous parts: a data source (usually a search engine wrapper), preprocessing (stemming or

lemmatization, identifying sentence boundaries and irrelevant terms), finally clustering and visualization. Many of these remain very similar from system to system and it would be a waste of time and effort to rewrite them. In Carrot^2 we have physically split these functional areas into separate *components*, so that almost any part of the system can be easily reused or replaced with another implementation or algorithm.

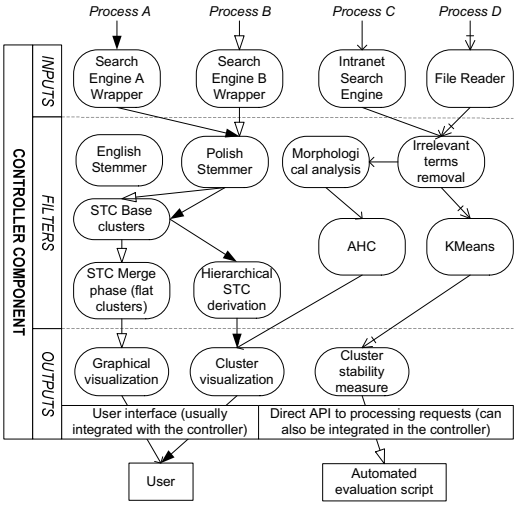

Fig. 1. Sample potential components and their arrangement into four processing chains in Carrot^2 framework. Processes are distinguished with different arrowheads. We show how different algorithms and input components can be reused to achieve different results

We have distinguished the following types of components: *inputs*, *filters*, *outputs* and *controllers*. Inputs accept a query and produce an initial list of references to documents. Filters alter the result by pruning, filtering, term conflation or adding additional information like that about clusters or suggested query expansion keywords. Outputs accept the filtered result and produce a response (whether visual output to the user or some other response is up to the component designer). Finally, a special type of component is the *controller*. It is in charge of interacting with the environment, for example, displaying a search interface to the user and displaying whatever the output component returned as a result. It also manages data exchange among other parts of the system when a query is processed according to a certain *processing chain* – ordered sequence of components starting with an input, with filters in between and ending in some output. Components are data-driven, independent programs, which can be perceived as black boxes – only the specification of input and output data format is important, while inside processing is entirely up to the designer. Since the mentioned

ease of component programming was our main goal, we decided to refrain from the new wave of technologies like SOAP or XML-RPC, and remain with HTTP POST. This provides great flexibility in adding new components, and provides a very useful abstraction layer from the language of implementation and physical location of a component. Handling of POST requests is relatively easy, with a number of very good implementations (for almost any language). For ad hoc programming, even any web server exposing CGI interface and shell scripts can be used. If performance counts, the component may be easily rewritten to any other computationally efficient language.

In order for components to understand each other's response, a protocol, or data exchange format must be established. It must be stressed that this format is fixed only for input components (query format), whereas data passed further on depends solely on the domain the framework is used in. For example, in our application to search results clustering we specified an XML-based format for exchanging data about snippets, clusters and certain linguistic analysis. Other applications might define their own data exchange specifications, however a large part of the system (like usually the most complex controller component) still remains reusable. A clear and stable format of data exchange ensures that any conforming component may be replaced with other implementation without harming the rest of the system (see Fig. 1).

5 Experiments

In this section we present observations from an experimental application of STC to English and Polish, along with a short discussion of existing approaches to estimating the quality of results in search results clustering.

5.1 Cluster quality evaluation

Evaluation of results is perhaps the most difficult part in development of a new search results clustering algorithm. Questions about what constitutes good and what bad clustering remain unanswered. In the past, several approaches have been proposed. Standard IR techniques such as precision and recall have been commonly used [10, 6], but they usually require a reference set of documents, for which the clusters are already known. While such databases exist (TREC, OHSUMED, Reuters), their structure is unsuitable for the domain of search results clustering because of the number and lengths of documents. Besides, with predefined collections there is always a danger that the algorithm discovers more subtle, or even better arrangement of groups than humans. According to [12] and also evident in our experiment, humans are rarely consistent both in number, arrangement and structure of a "perfect" clustering. Nonetheless, various forms of comparisons to a predefined structure of clusters have been used [1, 10, 6]. Especially the measure of cluster quality based on *information-theoretic entropy* [13], used in [1], seems to catch the balance between similarity of the reference set and the clusters being compared. Of course a single value

cannot represent all aspects of clustering quality. In our experiments we decided to employ it anyway in order to catch the *trends* and overall view of clusters structure with respect to changing input language and algorithm thresholds, rather than say whether the results are good or bad. It may even be the case that a confident measure of quality will be difficult to express in a mathematical formula; instead an empirical testimony of algorithm's usefulness to final users can be taken as a proof of its value. This approach has also been investigated and used. In [10] and [2], authors compare a standard ranked-list user interface to the clustered one using log analysis from real working systems. Also, explicit surveys have been used to measure users' level of satisfaction. These methods, while providing an excellent feedback from real people, have the drawback of being error prone due to subjectiveness of participants, their ranging level of patience and experience with Web searching.

Interestingly, the first thing the user sees as a results of a clustering algorithm – cluster descriptions – has never been a serious criterion of quality analysis. As suggested by Raul Valdes-Perez, president of a commercial search clustering system – Vivisimo – factors such as label conciseness, accuracy and distinctiveness could be taken into account together with the overall structure of the discovered clusters. In our opinion even light morphological analysis could help greatly in discriminating good (or at least grammatically correct) cluster labels from bad ones.

In our experiments we decided to use a normalized version of Byron Dom's entropy measure, briefly presented in Def. 1. Value of zero denotes no correspondence between the measured clusters and the ground truth set, one means the two are identical.

Definition 1. *Let X be a set of feature vectors, representing objects to be clustered, C be a set of class labels, representing the desired, optimal classification (also called a ground truth set), and K be a set of cluster labels assigned to elements of X as a result of an algorithm.*

Knowing X and K one can calculate a two-dimensional contingency matrix $H \equiv \{h(c,k)\}$, where $h(c,k)$ is the number of objects labeled class c that are assigned to cluster k. Information-theoretic external cluster-validity measure is defined by:

$$Q_0 = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{h(c,k)}{n}\log\frac{h(c,k)}{h(k)} + \frac{1}{n}\sum_{k=1}^{|K|}\log\binom{h(k)+|C|-1}{|C|-1}. \qquad (1)$$

where $n = |X|$, $h(c) = \sum_k h(c,k)$, $h(k) = \sum_c h(c,k)$.

Byron Dom's measure is defined for flat partitioning of an input set of objects, while STC produces flat, but overlapping clusters. In order to be able to apply the measure, we decided that snippets assigned to more than one cluster would belong only to the one having the highest score assigned by STC.

5.2 Test data and results

As a test data set for the experiment we collected real search results (the first 100 hits) to 4 English and 4 Polish queries. Multi-topic queries were selected so that a structure of clusters definitely existed: *salsa*, *intelligence*, *merced*, *logic*, *knowledge discovery* and certain subqueries such as *salsa sauce*, *merced lake* or *fuzzy logic*. This data was then clustered manually and independently by 4 individuals. The final ground truth set of clusters for each query was a result of experts discussion and unification of their results. For each query we applied STC in several configurations of input data preprocessing – with and without stemming, lemmatization and stop words removal. We used Porter stemmer for English and our own quasi-stemmer and dictionary lemmatization for Polish [3]. We also ran the algorithm over a wide range of control thresholds. Then we compared the generated clusters to the ground truth set using the above measure of quality.

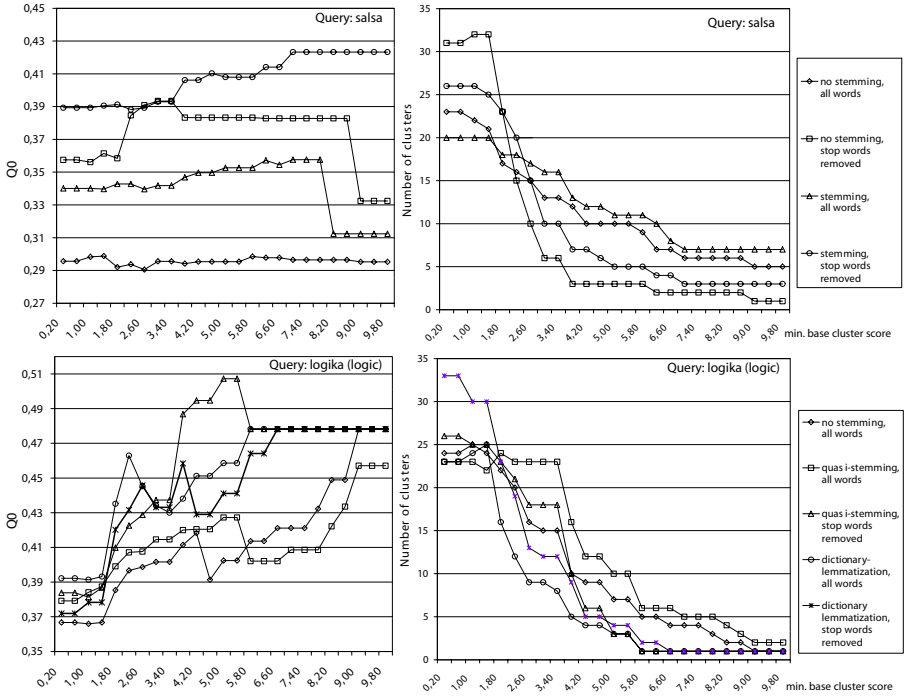

Fig. 2. Quality measure (left) and number of clusters (right) with respect to changing base cluster score and various configurations of input data pre-processing

The positive influence of language pre-processing has been confirmed with reference to our preliminary studies in [3]. Both term conflation algorithms and stop words removal were increasing the chosen measure of quality. However, the

gain was not always as optimistic as it can be observed on the *salsa* query in Fig. 2 Sometimes, both in the case of Polish queries (see *logika*), and English queries, the "winning" configuration was not the one with full input data treatment (although it always ranked higher than no pre-processing at all). We expected a much clearer difference in quality between raw and processed input data in Polish because of its rich inflection. Intuitively, without term conflation inflected phrases in Polish will not construct a single base cluster by STC and thus should decrease the overall similarity to the ground truth set. In the light of our analysis it is quite difficult to justify this expectation – the quality distribution over different pre-processing configurations was similar for both analyzed languages. Perhaps a more advanced lemmatization algorithm would bring some noticeable improvements.

As for the significance of STC thresholds, due to the limited space in this paper we cannot graphically depict their distribution. However, we noticed that merge threshold is only slightly affecting the quality of the discovered clusters (similar observation was given in [2]). The minimal base cluster score had a much greater impact as there was a clear relationship between its value and the number of produced clusters. This is natural as it is a cut-off threshold for the merge phase, but there was always a visible and steep drop in the number of clusters around a certain value of this threshold (see Fig. 2). This tendency was translated into a slight, but observable increase in the quality measure. This indicates, that one must be very careful in selecting base cluster score threshold, because even a small change of its value may severely affect the number of discovered clusters.

6 Conclusions

Carrot^2 is characterized by several features, which we think are useful for creating custom web search clustering systems. It defines a clear data-driven process of information manipulation. It decreases the time and effort required for building software and allows performing rapid experiments with search results clustering through component reuse and facilitation of their development. The project is provided as open source[2] and aimed mostly at the research community.

We have performed an experimental evaluation of the impact that the pre-processing of two different languages has on the performance of a representative search results clustering algorithm (STC). According to our best knowledge, such experiments have not been performed so far. As it appears from the results, careful language pre-processing does positively influence the quality of discovered clusters (with respect to the entropy-based measure we used). However, it is difficult to evaluate which of the used pre-processing methods (stemming, lemmatization, stop words removal) had the greatest impact on the measured quality. It is also interesting that in case of Polish, a much simpler technique of term conflation (quasi-stemmer) yielded similar results to the dictionary lemmatization method. We observed that the choice of base cluster score threshold of

[2] http://www.cs.put.poznan.pl/dweiss/carrot

the STC algorithm is a crucial issue as it strongly affects the number of discovered clusters.

Our experimental study does not limit the possibilities in measuring clustering quality. In particular, we would like to employ the grammatical structure of cluster labels and use this information to check whether they are understandable and informative to the user.

In terms of Carrot^2's development, we would like to focus on development of new components for linguistic analysis (stemming, lemmatization) and search results clustering algorithms (hierarchical methods in particular).

Acknowledgement. This research has been supported by grant 91-393/03-DS.

References

1. Maarek, Y.S., Fagin, R., Ben-Shaul, I.Z., Pelleg, D.: Ephemeral document clustering for web applications. Technical Report RJ 10186, IBM Research (2000)
2. Zamir, O., Etzioni, O.: Grouper: a dynamic clustering interface to Web search results. Computer Networks (Amsterdam, Netherlands: 1999) **31** (1999) 1361–1374
3. Weiss, D., Stefanowski, J.: Web search results clustering in Polish: Experimental evaluation of Carrot. Accepted for New Trends in Intelligent Information Processing and Web Mining Conference, Zakopane, Poland (2003)
4. Everitt, B.S., Landau, S., Leese, M.: Cluster Analysis. fourth edn. Oxford University Press (2001)
5. Hearst, M.A., Pedersen, J.O.: Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In: Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval, Zürich, CH (1996) 76–84
6. Boley, D., Gini, M., Gross, R., Han, S., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J.: Partitioning-based clustering for web document categorization. Decision Support Systems **27** (1999) 329–341
7. Hannappel, P., Klapsing, R., Neumann, G.: MSEEC - a multi search engine with multiple clustering. In: Proceedings of the 99 Information Resources Management Association International Conference, Hershey, Pennsylvania (1999)
8. Masłowska, I., Słowiński, R.: Hierarchical clustering of large text corpora. Accepted for New Trends in Intelligent Information Processing and Web Mining Conference, Zakopane, Poland (2003)
9. Dong, Z.: Towards Web Information Clustering. PhD thesis, Southeast University, Nanjing, China (2002)
10. Zamir, O.: Clustering Web Documents: A Phrase-Based Method for Grouping Search Engine Results. PhD thesis, University of Washington (1999)
11. Larsson, J.N.: Structures of String Matching and Data Compression. PhD thesis, Department of Comp. Science, Lund University (1999)
12. Macskassy, S.A., Banerjee, A., Davison, B.D., Hirsh, H.: Human performance on clustering web pages: A preliminary study. In: Knowledge Discovery and Data Mining. (1998) 264–268
13. Dom, B.E.: An information-theoretic external cluster-validity measure. Technical Report IBM Research Report RJ 10219, IBM (2001)