An Efficient Technique to Improve Snippet Clustering and Labeling using Modified FPF Algorithm

M. Hanumanthappa
Phd, Professor, Department of Computer Science & Applications, Bangalore University, Bangalore.

B R Prakash
Research Scholar, Department of Computer Science & Applications, Bangalore University, Bangalore

ABSTRACT

Document clustering is an effective tool to manage information overload. By grouping similar documents together, we enable a human observer to quickly browse large document collections, make it possible to easily grasp the distinct topics and subtopics. In this Paper we survey the most important problems and techniques related to text information retrieval: document pre-processing and filtering, word sense disambiguation, Further we present text clustering using Modified FPF algorithm and comparison of our clustering algorithms against FPF, which is the most used algorithm in the text clustering context. Further we introduce the problem of cluster labeling: Cluster labeling is achieved by combining intra-cluster and inter-cluster term extraction based on a variant of the information gain measure.

Keywords:

Clustering, document clustering, Cluster Labeling, Information retrieval.

1 . INTRODUCTION

The invaluable tool for retrieving information from the Web is the Search engines. The search engines return a list of results ranked in order, according to the query posed by the user. The user starts examining the result from top of the list and follows it down until the required information is found. The real difficulty is that the user cannot exactly figure out what is the relevant information, because the query may be short and their interpretation might be ambiguous in the absence of context. In contrast another approach to web information retrieval is based on categorization, and letting the user to see the results associated with categories that best matches the user query. But this covers only a small fraction of the existing pages[1,2]. Now the leading approach is search results clustering, depending on the user query the search engines present top ranked list of the web snippets. Sometimes the snippet quality also will be an important issue, because it allows the user to decide whether that is the required page which matches the information needed. A good-quality snippet is one which includes the page title, text fragment (short), file type, date/year etc,. in some snippets we can expect query terms also. The items that are obtained from various search engines integrated by the Meta- Search Engine in order to increase the coverage of the results. MSEs are used in order get large set of results. But the drawback of using MSEs is difficulty in using effective algorithms for merging the ranked lists results obtained by various other search engines [3].

Clustering avoid showing the user some of the equivalent documents in the first page of results. This is one of the advantages of regular search engine which does the clustering. This activity is close to the classical duplicate or near-duplicate detection in information retrieval, and one can take advantage of the fact that duplicates and near duplicate are easy to detect via multiple hashing or shingling techniques, and these tasks can be carried out, at least in part, in an off-line setting, but duplicate detection activity labeling is not an issue.

The way of organizing the snippets into a set of meaningful groups returned by a search engine in response to a query is Search Results Query.

2. PHASES OF SEARCH RESULTS CLUSTERING

The main phases of the search results clustering process that is each phase is highlighted more on text processing techniques and is as follows[4]:

2.1 Phase-1 Snippet acquisition

This phase collects document snippets that have been returned by a search engine with respect to the user's query. For web search results clustering the snippets can be obtained directly from a web search engine using an appropriate API, such as Google API6, or by parsing the engine's HTML output and extracting all necessary data.

2.2 Phase-2 Preprocessing

The preprocessing phase is to transform the raw text of snippets to the form suitable for the clustering algorithm. This phase usually involves the following text processing techniques:

a. **Filtering** — some special characters like '%', '$' or '#', etc are to be filtered out. Because these kinds of characters would introduce noise and affect the quality of clustering. So in this phase such characters have to be removed from the input snippets.
b. **Tokenization** — the process of identifying word and sentence boundaries in a text is Tokenization. The tokenizer may be white space characters as word delimiters and selected punctuation marks such as '.', '?' and '!' as sentence delimiters. But this approach is not robust to tackle difficult content such as document snippets. More elaborate tokenization techniques have to be applied.
c. **Stemming** — a portion of a word that is left after removing its suffixes and prefixes is a stem. While stemming all words in a text are replaced with their respective stems. With the use of a stemmer (short

for a stemming algorithm) different grammatical forms of a word can be reduced to one base form. For example, the words: abus, abuse, abused, abuser, abuses, abusing should be transformed to the word abuse.

d. **Stop word removal —** Stop word removal is a process of identifying and removing stop words from a text. Stop words, also referred to as function words, are words that have no identifiable meaning and are of little use in text processing tasks. For English, stop words are among others auxiliary verbs, such as have, be, pronouns, such as he, it or prepositions, such as to and for.

e. **Language recognition —** Stop word removal, stemming and spell checking make an implicit assumption that the language of the processed text is known. While it may be the case with relatively small collections of documents, in general not only does the language remain unknown, but also it is rarely stated explicitly in the text. Thus, language recognition is an important part of a search results clustering system.

2.3 Phase-3 Feature selection

This phase is to identify words in a text that are non-informative according to corpus statistics and can be omitted during clustering. The reasons for using feature selection in search results clustering are twofold. First of all, in most cases limiting the number of features considerably increases the time efficiency of the clustering algorithm. Secondly, feature selection can help to remove noise from a text, which may result in higher accuracy of clustering.
Some of feature selection strategies that are suitable for the clustering problem are reviewed and compared:

a. Document Frequency (DF) — This method selects only those words that appear in more than a given number of documents. This is simple and efficient selection strategy in a natural way that scales to large number of documents.

b. Term Strength (TS) — Term Strength is computed for pairs of documents for which the similarity measure exceeds a predefined threshold. For a particular term, Term Strength is the conditional probability that this term occurs in one document, given that it occurs in the other document. Because the probabilities may need to be calculated for all possible pairs of documents, TS computational complexity is quadratic with respect to the number of documents.

c. Term Contribution (TC) — One of the disadvantage of DF is that it favors terms that have a high occurrence frequency, not considering the term's distribution among different classes. The Term Contribution strategy alleviates this problem by aggregating the term's contribution to document similarity.

2.4 Phase-4 Clustering

The last phase in the process chain is the actual clustering algorithm. Several classes of algorithms have been used for the search results clustering task, ranging from adaptations of the classic numerical approaches, such as K-Means in Scatter/Gather, to purpose-built methods such as STC and SHOC[5][12].
FPF (Furthest Point First) is a polynomial algorithm for the clustering objective of minimizing the maximum intra cluster distance. When the weight function satisfies the triangle inequality, it guarantees a 2-approximation. The FPF works in iterations[6][7]. In each iteration it keeps a subset of nodes called heads and partitions V into clusters according to their distance from the heads. In each iteration the algorithm will choose the furthest point from the current set of heads as the new head.

FPF Algorithm:

- **Initialization:**

  1. Pick an arbitrary point, mark it as head1 - the head of cluster 1.

  2. Assign all points to cluster 1.

- **Iteration i (i = 2,,,,,,,,,,,,,,,,, k)**

  1. Pick the furthest point from the current set of heads.

  2. Designate it head i.

  3. Move to cluster i every point closer to head i than to its current head.

Furthest Point First algorithm has to be improved from both the computational point of view and the output clustering quality. No changes have been made with respect to computation but regarding quality some of the part of the clustering schema is modified. Using random sampling technique the clustering output quality is improved and this algorithm is called as M-FPF algorithm. One more virtue of FPF is that it chooses a set of centers not representative of the clusters. When FPF creates a new center, it picks the furthest point from the previous selected centers and thus the new center can likely be close to a boundary of the subspace containing the data set. In order to conquer this problem modify M-FPF was introduced in which medoids is used instead of centers.

Improvement with respect to computational point of view

The running time of the FPF algorithm is dedicated to compute distances in finding the closest center to each point. At the beginning of each iteration we have the distance of every point from its head. We search for the maximum distance in O(n) to find the new head. Then we compare for each point's the distance from its current head to the distance from the new head and update the point head if necessary. Therefore there is O(n) work in each iteration. Since there are k iterations the time complexity of FPF is O(kn). Unfortunately the worst case time complexity still remain O(kn) because the number of saved distance computations depends on data distribution and thus, it can not be predicted in advance. The modifications discussed here do not change the FPF output.

The modified algorithm is as follows:

Consider, in the FPF algorithm, any center Ci and its associated set of closest points Ci. Store Ci as a ranked list, in order of decreasing distance to ci. When a new center cj is selected, scan Ci in decreasing order of distance, and stop

scanning when, for a point p Є Ci, it is the case that d(p, ci) ≤ 1 2d(cj , ci).
This rule filters out from the scan points whose neighbor cannot possibly be cj , thus significantly speeding up the identification of neighbors. Note that all distances between pairs of centers must be available; this implies an added O(k2) cost for computing and maintaining these distances.

Improvement of output clustering quality using random sampling technique

The algorithm efficiency is improved by applying FPF algorithm. Instead of applying for random sample than whole data set using random sampling technique of random sample of size n'= $\sqrt{nk}$ of the input points. For all k≤n it is always true that n' ≤ n. It is possible to add the remaining (n - n') points to the cluster of their closest center. But again the problem is the time spent computing distances to the point to the closest center, so M-FPF is referred

```
M-FPF:
Data: Let O be the input set, k the number of desired clusters
      ∈
Result: C: a k-partition of O
Initialize R with a random sample of size  $\sqrt{|o|K}$
elements of O;
C= FPF(R,k);
forall C_i ∈ C do
| μ_i = getCenter(C_i);8
end
forall p in O \ R do
| assign p to cluster C_i such that  $d(p,\mu_j),\forall j \neq i;$
End
```

2.5 Text Clustering:

With respect to text clustering here we are comparing FPF and M-FPF. Comparison is done based on two different metric spaces: Cosine Similarity(CS) and Generalized Jaccard Coefficient(GJC). Data set for snippets is retrieved using Carrot2[10]. The data set what ever obtained with a manual classification from Carrot2, was the ground truth. Four measures are used to validate the clustering correspondence with this ground truth, the Normalized Mutual information(NMI), the Normalized Complementary Entropy(NCE), the F-measure and the Accuracy. All these measures are in the range [0, log n] i.e.,[0,1]where higher values mean better quality.

The below table represents a comparison of the two clustering algorithms. Each clustering was run using two different distance functions with the goal of comparing also the metrics for text clustering.

Dataset	Measure	FPF		M-FPF	
		CS	GJC	CS	GJC
Snippets	NMI	0.697	0.683	0.414	0.411
	NCE	0.428	0.407	0.678	0.687
	F-measure	0.360	0.395	0.338	0.350
	Accuracy	0.568	0.565	0.542	0.560

Table 1: comparison of FPF and M-FPF on the snippets with Generalized Jaccard Coefficient and Cosine Similarity

The F-measure and accuracy are always comparable; NMI, in the case FPF, is quite higher than that of M-FPF. In contrast NCE of FPF is lower than that of the other algorithms.

3. CLUSTER LABELING:

After a large amount of documents has been clustered, it is important to have as short description of its content other wise it is not useful. A cluster label should have some properties like, shortness, syntactically correctness and predictiveness[8][9]. As already mentioned above phases of search clustering algorithm. The FPF algorithm works in three phases –
Phase 1 – extracting a list of topic representative keywords for each cluster,
Phase 2 – removes some duplicate keywords according with a global criterion,
Phase 3 – extract the best possible sentence from the cluster such that it matches the cluster keywords

3.1 Extracting cluster Keywords

Selection of certain number of descriptive keywords for a particular cluster. The more a word is relevant for a document will have the highest score. By this the cluster can be synthesized by the set of words with the highest score. This set of words are called as candidate words. The algorithm is limited not to take the content of the other clusters. The selection of related clusters is referred to as local candidate selection.

3.2 Candidate word selection

The objective here is to decide for which cluster a shared keyword is more appropriate. This can be coined as a problem of classification in which each set of candidate words is a category and the keywords that appear in more than set of candidate words must be classified. At the end for each keyword we have an associated set, from which we can remove the word from the other sets.In order to classify candidate words a modified version of the information gain function is used.
Let x be a document taken uniformly at random in the corpus. P(t) is the probability that x contains term t, P(c) is the probability that x is in category c. The complementary events are denoted p($\bar{t}$)=1-(t) and P($\bar{c}$)= 1-(c). P(t,c) is the probability that s is in category c and contains term t, $p(\bar{t},\bar{c})$ is the probability x does not contain t and is not in category c. $p(t,c)$ is the probability that x contains t but is not contain t but is not in category c. $P(\bar{t}+c)$ is the probability that x does not contain t and s in category c. Clearly being these mutually disjoint events it holds:

$$P(t,c)+P(\bar{t},\bar{c})+P(t,\bar{c})+P(\bar{t}+c)=1$$

The information gain is:

$$IG(t,c)=\sum_{a\in\{t,\bar{t}\}}\sum_{b\in\{c,\bar{c}\}}P(a,b)\log\frac{P(a,b)}{P(a)p(b)}$$

IG measures the amount of information that each argument contains about the other; when t and c are independent, IG(t, c) = 0. If IG(t, c) is high, the presence or absence of a term t is deemed to be highly indicative of the membership or non-membership in a category c of the document containing it.
Information gain formula is the summation of four contributions: the presence of the term in the class

$(P(t,c)\log\frac{P(t,c)}{P(t)P(c)})$, the absence of the term in the other classes $(P(\bar{t},\bar{c})\log\frac{p(\bar{t},\bar{c})}{p(\bar{t})p(\bar{c})})$ the term in the other classes $(P(t,\bar{c})\log\frac{p(t,\bar{c})}{p(t)p(\bar{c})})$ and the absence of the term in the class $(P(\bar{t},c)\log\frac{p(\bar{t},c)}{p(\bar{t})p(c)})$

The first two contributions represent the “positive correlation” between the arguments while the last two factors represent their “negative correlation”.

Here the more concentration is focused about the presence of keyword in a candidate set to the detriment of the others and hence we are interested to those terms in information gain formula which positively describes the contents of a cluster. With respect to the above consideration information gain has been modified to :

$$IG_m(t,c) = (P(t,c)\log\frac{P(t,c)}{P(t)P(c)}) + (P(\bar{t},\bar{c})\log\frac{p(\bar{t},\bar{c})}{p(\bar{t})p(\bar{c})})$$

3.3 Label generation

The main objective of label generation is to extract from each cluster the most possible descriptive short sentence based on the candidate words. So we can select the sentences contained in the documents from the considered cluster, as a text corpus, we can arrange them in a inverted index and candidate words can be used as query in order to retrieve related sentences. Depending upon the phrase, labels are extracted from the top in the rank list. The linear scan of all the sentences is followed[11].

The ranking function will consider three parameters: weight of the candidate words present in the phrase the number of different candidate words retrieved their inter-distance

Let q = {q1, . . . qh} be the words of the query that produces the snippets corpus, C = {c1, . . .ck} the candidate keywords for the cluster W = {w1, . . . ,wn} the words in the considered window (in our case n = 5 and k = 3) such that wi has score s(wi), the ranking of W is: $R(w) = \sum_{w_i \in W} R(w_i)$

Where $R(w)$ is defined as:

$$R(w_i) = \begin{cases} s(w_i)\ if\ \ w_i \in C\ \forall\ j \langle\ i\ \ w_j \neq w_j \\ 0 \qquad if\ \ w_i \in Q\ \forall\ j \langle\ i\ \ w_j \neq w_j \\ -1/2\ if\ \exists\ j \langle\ i\ \ w_j = w_j \\ -1\ \ otherwise \end{cases}$$

The window with highest value of R is the candidate label.
Candidate labels are filter based on the following constraints:

- a word or a pair of words is removed in case of consecutive copies;
- similarly to what is made for stop words, a list of inadmissible “last word” is used to filter out, among the other: prepositions, conjunctions and articles;
- in the case in which the query is formed by a pair of words Q = {q1, q2}, if the label has q2 as first word, q1 is inserted before, instead, if q1 is the last word of the label q2 is appended;
- in case the same word is the initial and final word of the label an instance is removed (more precisely we remove the last instance if it is not preceded by an inadmissible last word, otherwise we remove the first instance),
- web URLs and e-mail addresses are removed. All the filter rules are recursively applied until the label reaches a stable form. The output of the filter is the final label. The quality of the resulting labels depends strictly from the quality of the sentences in the corpus and the clustering algorithm.

4. CONCLUSION

In this paper we describes FPF, a meta-search engine that groups into disjoint labeled clusters the Web snippets returned by auxiliary search engines. The cluster labels generated by M-FPF provide the user with a compact guide to assessing the relevance of each cluster to the information need. Striking the right balance between running time and
cluster well-formed ness was a key point considered. Both the clustering and the labeling tasks are performed on the processing only the snippets provided by the auxiliary search engines, Further research is needed in two main areas. First, we plan to assess to what extent of external knowledge can improve the system's performance without speed penalties. Second, it is possible to introduce in the current pipeline (input snippets are clustered, candidates are extracted, labels are generated) of the architecture a feedback loop by considering the extracted candidates labels as predefined categories, thus examining which snippets in different clusters are closer to the generated labels.

5. REFERENCES

[1] Geraci, F., Pellegrini, M., Sebastiani, F., Maggini, M.: Cluster generation and cluster labeling for web snippets: A ast and accurate hierarchical solution. Technical Report IIT TR-1/2006, Institute for Informatics and Telematics of CNR (2006)

[2] Nearest-neighbor searching and metric space dimensions. In Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors, Nearest-Neighbor Methods for Learning and Vision: Theory and Practice, pages 15–59. MIT Press, 2006.

[3] Flavio Chierichetti, Alessandro Panconesi, Prabhakar Raghavan, Mauro Sozio, Alessandro Tiberi, and Eli Upfal. Finding near neighbors through cluster pruning. In Proceedings of ACM PODS, 2008.

[4] Paolo Ferragina and Antonio Gulli. A personalized search engine based on Web-snippet hierarchical clustering. In Special Interest Tracks and Poster Proceedings of WWW-05, 14th International Conference on the World Wide Web, pages 801–810, Chiba, JP, 2006

[5] Karina Figueroa, Edgar Ch´avez, Gonzalo Navarro, and Rodrigo Paredes. On the least cost for proximity searching in metric spaces. In 5th International Workshop on Experimental Algorithms (WEA), volume 4007 of Lecture Notes in Computer Science, pages 279–290. Springer, 2006.

[6] M. Furini, F. Geraci, M. Montangero, and M. Pellegrini. VISTO: VIsual Storyboard forWeb Video Browsing. In CIVR ’07: Proceedings of the ACM International Conference on Image and Video Retrieval, July 2007.

[7] F. Geraci, M. Pellegrini, F. Sebastiani, and M. Maggini. Cluster generation and cluster labelling for web snippets. In Proceedings of the 13th Symposium on String Processing and Information Retrieval (SPIRE 2006), pages 25–36, Glasgow, UK., October 2006. Volume 4209 in LNCS.

[8] Filippo Geraci, Marco Pellegrini, Paolo Pisati, and Fabrizio Sebastiani. A scalable algorithm for high-quality clustering of Web snippets. In Proceedings of SAC-06, 21st ACM Symposium on Applied Computing, pages 1058–1062, Dijon, FR, 2007.

[9] Filippo Geraci, Mauro Leoncini, Manuela Montangero, Marco Pellegrini, and M. Elena Renda. Fpf-sb: a scalable algorithm for microarray gene expression data clustering. In Proceedings of 1st International Conference on Digital Human Modeling, 2008.

[10] Stanislaw Osinski and Dawid Weiss. Conceptual clustering using Lingo algorithm: Evaluation on Open Directory Project data. In Proceedings of IIPWM-04, 5th Conference on Intelligent Information Processing and Web Mining, pages 369–377, Zakopane, PL, 2004.

[11] D. Crabtree, X. Gao, and P. Andreae, "Standardized evaluation method for web clustering results," in Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, 2005.

[12] Montserrat Mateos Sánchez, Encarnación Beato Gutiérrez,Roberto Berjón Gallinas, Ana Mª Fermoso García, Miguel Angel Sánchez Vidales CLUSTERING OF WEB DOCUMENTS: FULL-TEXT OR SNIPPET? IADIS International Conference WWW/Internet 2008, pp488-493. ISBN: 978-972-8924-68-3 © 2008 IADIS.