MONTE CARLO PROJECTIVE CLUSTERING OF TEXTS

Vladimír Ljubopytnov and Jaroslav Pokorný
Department of Software Engineering, Faculty of Mathematics and Physics, Charles University, Praha, Czech Republic

Keywords: Document clustering, Projective cluster, Web snippets, Web mining, Monte Carlo method.

Abstract: In this paper we propose a new, improved version of a Monte Carlo projective clustering algorithm – DOC. DOC was designed for general vector data and we extend it to deal with variable dimension significance and use it in web search snippets clustering. We discuss advantages and weaknesses of our approach with respect to known algorithms.

1 INTRODUCTION

Document or text clustering is a field of Information Retrieval (IR) (Manning, Raghavan, and Schütze, 2008) that optimizes search in custom document collections as well as World Wide Web. There are collections of high-dimensional data (and texts are among them) that inherently tend to group in subspaces. It is common in IR to store texts as vectors of numbers indicating the presence or the number of terms (vector space model of IR). The dimensions of these vectors correspond to index terms (words or phrases) and subspaces shared by multiple texts may define a distinctive and interesting topic. Unfortunately, clustering suffers from the curse of dimensionality. This is the motivation behind using a projective clustering algorithm.

We chose DOC, a scalable Monte Carlo method for clustering coordinate vectors introduced by Procopiuc (Procopiuc et al, 2002). It is linear in the number entities to be clustered and polynomial in dimension (the exponent can be changed to accommodate high-dimension data more easily). Subspaces recognized by DOC are perpendicular to the coordinate axes. In text clustering, information about dimension (word or phrase) significance is available. Thus, it made sense to extend the algorithm to include this data in cluster evaluation.

One can observe that clustering of the results returned by search engines becomes prevailing in recent times. The goal of our research is to develop a more efficient clustering Web snippets.

In Section 2 we give a short overview of the state-of-art algorithms for document clustering, particularly for cases in which documents are Web snippets. Then in Section 3 we introduce the basic DOC algorithm and present our variant – DocDOC with an application on Web snippets. We also discuss possibilities of its usage. Finally, we discuss its position among existing approaches and mention some other possibilities of its application.

2 RELATED WORKS

Today search engines return with a ranked list of search results also some contextual information, in the form of a Web page excerpt, the so called snippet. Web-snippet clustering is an innovative approach to help users in searching the Web. It consists of clustering the snippets returned by a (meta-) search engine into a hierarchy of folders which are labelled with a term. The term expresses latent semantics of the folder and of the corresponding Web pages contained in the folder. The folder labels vary from a bag of words to variable-length sentences. On the other hand, snippets are often hardly representative of the whole document content, and this may in some cases seriously worsen the quality of the clusters.

Remind that search engine result summarizations is a subcategory of Web content mining. There are various approaches to snippets clustering in literature. Zamir and Etzioni presented the Suffix Tree Clustering (STC) algorithm (Zamir and Etzioni, 1998). It uses suffix tree to identify groups of documents sharing a common phrase. These groups are merged if their overlap exceeds given threshold using a single link method. The phrase awareness of

Ljubopytnov V. and Pokorný J. (2009).
MONTE CARLO PROJECTIVE CLUSTERING OF TEXTS.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 237-242
DOI: 10.5220/0002247602370242

this algorithm caused a positive jump in quality when compared to classical algorithms. However, it may suffer from the chaining effect of the single linkage in some cases and the cluster scoring is based on phrase length and cluster size only.

In 2003, Maslowska introduced Hierarchical STC, which replaces single link merging with topological ordering of a directed cluster graph (Maslowska, 2003). The edges between base clusters are defined by inclusion ratio of respective base clusters and again, a threshold ratio is given. Fully overlapping base clusters are merged in advance. The result is a hierarchy, where those clusters that are not included by any other are on top.

The Lingo algorithm (Osinski, Stefanowski, and Weiss, 2004) uses singular vector decomposition (SVD) to find correlating index term groups and selects best among them while preferring phrases. Phrases are discovered efficiently using a suffix array. Roughly said, found phrases are used to create cluster labels and documents are assigned to their best matching label. This approach (description-first) concentrates on label quality which suffers in classical algorithms. It is designed for web snippet clustering where the time costly SVD does not show itself.

A more advanced algorithm is described in (Mecca, Raunich, and Pappalardo, 2007). Its main contribution is a novel strategy – called dynamic SVD clustering – to discover the optimal number of singular values to be used for clustering purposes. Authors apply the algorithm also on the full documents and justify the idea that clustering based on snippets has inherently lower quality than on full documents. The algorithm was used in the Noodles system - a clustering engine for Web and desktop searches.

Very efficient online snippet clustering based on directed probability Graphs is described in (Li and Yao, 2006)

Web-snippet clustering methods are usually classified in according to two dimensions: words vs. terms and flat vs. hierarchical (Ferragin and Gulli, 2006). Four categories of approaches are distinguished:

1. Word-Based and Flat Clustering
2. Term-Based and Flat Clustering
3. Word-Based and Hierarchical Clustering
4. Term-Based and Hierarchical Clustering

Our considerations concern rather the second category of methods. A lot of background material to clustering, including web snippets, can be found in (Húsek et al, 2007).

3 DocDOC CLUSTERING ALGORITHM

In this chapter we describe the original DOC algorithm and its variant appropriate for Web snippets.

3.1 DOC Algorithm

The DOC algorithm operates with projective clusters that additionally store information about bound dimensions (dimensions where all vectors are sufficiently near with respect to a pre-defined value). When looking for a cluster, it randomly guesses a combination of vectors, finds their bound dimensions, and returns all vectors that share bound dimensions with those randomly selected. Multiple guesses are made to ensure optimality of returned cluster with proven probability. A key feature of this algorithm is the use of special quality function that takes cluster size as well as bound dimensions into account. This function is called *β-balanced*. In other words, a β-balanced function is any function

$$\mu: R \times R \rightarrow R$$

increasing in both parameters, that fulfils two conditions:

1. $\mu(0, 0) = 0$
2. $\mu(\beta a, b + 1) = \mu(a, b)$

for $0 \leq \beta < 1$ and any non-negative a and b. For example, the function defined as

$$\mu(a, b) = a(1/\beta)^b$$

is β-balanced.

When applied to cluster size $|C|$ and dimensionality $|D|$ it returns the same quality as for smaller cluster with size $\beta|C|$ and dimensionality increased by one ($|D|+1$). β specifies a trade-off between cluster size and dimension and is one of algorithm parameters. The other – α – sets minimal recognized relative cluster size. DOC algorithm is proven to return an optimal projective cluster (or an approximation of it) with probability at least 50%, if $1/4n \leq \beta < 1/2$ (n denotes total number of dimensions) and $0 \leq \alpha < 1$. For full detail see (Procopiuc et al, 2002).

3.2 Adapting DOC to Text

Here presented extension of DOC, called DocDOC, was introduced in (Ljubopytnov, 2006). When adapting DOC to web snippets, some obvious observations were done:

- In web search snippets, multiple term occurrences hardly influence term importance. Thus, term frequency (*tf*) was discarded (in contrast with widely used *tf*×*idf* metric), leaving only dimension bound weighing in play. This enables use of Boolean model with weights and efficient bit array representation.
- Only shared presence of term defines bound dimension as association power of non-present terms is next to zero. This makes determining bound dimensions a matter of bitwise AND operation.
- A change to the quality function that favours better weighed dimensions should be employed to make full use of significance estimates (*idf*, etc.).

First two points are trivial but the third step requires further explanation. We mentioned that β parameter is the trade-off between cluster size and dimension. Having dimensions of different value, it is this trade-off that should be set individually for each dimension to determine projective cluster quality. If we have a good, preferred dimension, it should be more costly to throw away. Fewer objects are required to stay in the cluster to keep its quality if we try to bind this highly priced dimension. Thus, we see that for better weighed dimensions we want smaller β.

In the DocDOC algorithm, dimension weights are transformed to fit into the interval $[1/4n; 1/2)$ as required for the β value in DOC and stored in a function

$$\beta': D_{all} \rightarrow [1/4n; 1/2)$$

where D_{all} is set of all dimensions. *β'-balanced function* is any function

$$\mu : R \times P(D_{all}) \rightarrow R$$

increasing in both parameters (in case of second parameter, natural set ordering is assumed), that fulfils two conditions:

1. $\mu(0,\emptyset) = 0$
2. $\mu(\beta'(i)a, D \cup \{i\}) = \mu(a, D)$

for $0 \leq \beta < 1$ and any non-negative a and D subset of D_{all}. For example, the function defined as

$$\mu'(a, D) = a \prod_{i \in D} 1/\beta'(i)$$

is β' balanced.

Proof of correctness of this change is easy. The key is that we keep the β' values in required range as they vary across dimensions. Once we ensure this, the proof works identically as for original DOC in (Procopiuc et al, 2002).

Algorithm 1: DocDOC procedure.

```
Procedure DocDOC(P, dim_score, α, β)
1. r := log(2n)/ log(1/2β);
2. m := (2/α)^r ln 4;
3. hash := Ø;
4. best_quality := -1;
5. for i := 1 to 2/α do
    begin
    Randomly select p ∈ P;
    for j := 1 to m do
    begin
       Randomly select X ⊆ P, |X| = r;
       D := {i| |q_i - p_i| ≤ w, ∀q ∈ X};
       if D ∈ hash
       then quality = hash(D);
       else begin
        C := B_p,D ∩ P;
        quality := μ'(|C|,D,dim_score);
        hash(D) := quality;
            end
       if best_quality < quality
       then begin
           best_quality := quality;
           C* := C;
           D* := D;
            end
    end
   end
6. return (C*,D*)
```

DocDOC parameters remain the same, i.e. β is used to set the lower bound for β' values (best dimension β). The DocDOC Procedure is described by Algorithm 1.

3.3 Phrase Discovery

Lingo makes use of phrases and evaluates them using SVD (correlation strength). STC sets the phrase score according to its length. Although such pre-processing introduces additional time cost, we decided to use a more thorough phrase evaluation scheme based on the LLR-test (Log Likelihood Ratio) described, e.g. in (Cox and Hinkley, 1974).

We look at all bigrams and trigrams and we study the hypothesis of them being a random collocation (their word occurrences are independent). Using *G*-test, we determine the likelihood ratio of this hypothesis:

$$G = O_i \log(O_i/E_i),$$

where Oi is observed event frequency and Ei is expected event frequency according to the hypothesis, i iterates over all possible events. For a bigram w1w2 we have a contingency table in Table 1 describing a number of situations when w1 does or

Table 3: The best scored phrases for queries “salsa” and “clinton”.

”salsa” (912)	*score*	”clinton” (300)	*score*
salsa danc	18.84	clinton county	25.03
salsa congres	14.09	bill clinton	23.03
salsa recip	12.87	hillary clinton	21.82
salsa festival	12.06	clinton presidential	19.66
salsa lesson	11.71	new york	19.57
danc salsa	10.58	clinton porti	18.23
salsa verd	10.47	united stat	17.42
danc	10.43	barack obama	17.03
salsa music	10.43	clinton lead	16.39
music	10.26	clinton say	16.39
salsa class	9.85	clinton administration	16.39
latin	9.82	whit hous	16.26
salsa dancer	9.78	clinton hill	15.37
salsa club	9.75	clinton memorial	14.37
event	9.45	clinton jumped	14.16
recip	9.23	rodham clinton	14.16
salsa scen	8.99	clinton impeachment	14.16
latin danc	8.82	clinton st	14.16
cha cha	8.76	presidential rac	13.20
san francisco	8.40	chamber of commerc	13.16

does not precede w_2 and w_2 does or does not supersede w_1.

Table 1: Contingence table for a bigram w_1w_2.

	w_2	$\neg w_2$
w_1	n_{11}	n_{12}
$\neg w_1$	n_{21}	n_{22}

If we know the bigram w1w2 occurrence frequency c12, the occurrences of words w1: c1, w2: c2, and the total bigram count B we can write the contingency table as it is on Table 2.

Table 2: Improved contingence table for a bigram w_1w_2.

$$\begin{matrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{matrix} \begin{bmatrix} = & c_2 \end{bmatrix} - c_{12} \quad \begin{bmatrix} c_{12} & c_1 - c_{12} \\ B - c_1 - c_2 + c_{12} & \end{bmatrix}$$

For the independence hypothesis, expected event frequencies are

$$m_{ij} = \frac{n_{ip} * n_{pj}}{B}$$

where $n_{ip} = \Sigma_j n_{ij}$ and $n_{pj} = \Sigma_i n_{ij}$. We execute the *G*-test on these values:

$$G = \sum_{ij} n_{ij} \log(n_{ij} / m_{ij})$$

The higher the value, the more the probability of bigram being not random co-occurrence but a phrase.

For trigrams, the situation is more complicated as there can be as much as four models of trigram words independence (one for mutual independence of all words in trigram and three for independence of one word on other two that correlate). Contingency table is three dimensional and sub-bigram frequencies are needed for its construction. Hypothesis with best (lowest) score is returned (it is nearest to reality).

For longer *n*-grams, the complexity keeps growing. We settled with bigrams and trigrams only. If a frequent quadgram is present in the text, it’s first and second trigram will rank high and they will be associated together by DocDOC, but only in case, that there are no stoplist words at their boundaries. Otherwise, most high ranking phrases contain a stop word on its boundary, thus being quite useless (e.g., ”lot of ”, ”in the”).

Scores obtained by *G*-test tend to grow rapidly, so we logarithmized them to get a reasonable scale and combined with inverted document frequency.

Tests presented here have been performed with a sample of 912 snippets obtained by Google for the query “salsa” and 300 snippets for the query “clinton”. The results are presented in Table 3.

3.4 Optimizations

Web snippets demonstrate strong regularities; dimension (word, phrase) combinations appear

repetitively when trying to guess an optimal cluster. In contrast with original DOC, we use a hash structure to store previously calculated dimension qualities. That saves us a costly data scan per each cluster quality computation. Hash is indexed by used dimension combinations.

The benefit of dimension caching may be worth over 50% of data scans for web snippets. It grows if the DocDOC procedure is called multiple times on the same data (i.e. to bump the optimality probability). The beauty of dimension cache is also that when sorted, it gives a ranked list of overlapping clusters that is identical to STC base clusters, only with 100% overlapping clusters merged. This is very important, since that makes DocDOC identical to STC with respect to the result achieved (when identical postprocessing is done, i.e. hierarchization).

3.5 Usage and Output

The DOC procedure returns one cluster at a time. There are three possibilities how to use it:

- greedy – found clusters are removed from collection, this relies heavily on cluster quality estimation by the quality function, valuable information can be lost, must be run several times to increase the guaranteed 50%+ probability of returning an optimal
- cluster (empirically, the algorithm returns good clusters all the time, but potentially destroying the optimal one), runs while there is enough data (with collection size percentage as threshold) and its running time may vary substantially.
- overlapping – every found cluster is remembered in a list sorted by quality and the procedure is run few times. This improves result stability and speed, loses no information and can be dealt with as with merged STC base clusters, that is, merging and rescoring until no clusters overlap enough.

Final task is creating cluster labels from (*C*, *D*) projective clusters. That is done by choosing all or best ranking dimensions (phrases, words) from *D*. As in HSTC, clusters can be topologically sorted to get a nice cluster hierarchy.

4 COMPARISON OF ALGORITHMS

We have made the following observations:

- Clusters found by DOC are exactly those generated by STC, but merged in case of 100% overlap.
- Suffix arrays are 5 times more efficient with respect to memory usage than suffix tree.
- DocDOC has a potential for more flexible ranking than STC.
- The quality function introduced in DocDOC defines different ordering of clusters than STC.
- DocDOC is better parallelizable and scalable.
- DocDOC need less memory footprint than Lingo and maybe STC.
- The quality function introduced in DocDOC defines different ordering of clusters than STC.

5 CONCLUSIONS

We proposed and implemented an improved version of the DOC algorithm used on Web snippets in our experiments. We have shown that it has a number of better properties than other algorithms of this category.

Since discovering knowledge from and about Web is one of the basic abilities of an intelligent agent, an applicability of the algorithm can be found e.g. in semantic Web services.

Although named DocDOC, the algorithm has far greater usability than just texts. If used on vector data (as opposed to the Boolean model, but keeping the dimension weight information), there are applications across various disciplines (i.e. medicine, data mining) that may benefit from this algorithm.

ACKNOWLEDGEMENTS

This research was supported in part by GACR grant 201/09/0990.

REFERENCES

Cox, D. R. and Hinkley, D. V: Theoretical Statistics, Chapman and Hall, 1974.

Ferragin, P. and Gulli, A.: A personalized search engine based on Web-snippet hierarchical clustering. In: Proc. of 14th International Conference on World Wide Web 2005, Chiba, Japan, pp. 801-810.

Húsek, D., Pokorný, J., Řezanková, H., Snášel, V.: Data Clustering: From Documents to the Web. Chapter 1 in book Web Data Management Practices: Emerging Techniques and Technologies, A. Vakali and Pallis Eds., Idea Group Inc., 2007, p. 1-33.

Li, J., Yao, T.: An Efficient Token-based Approach for Web-Snippet Clustering. In: Proc, of the Second International Conference on Semantics, Knowledge, and Grid (SKG'06), 2006, 6 pages.

Ljubopytnov, V.: Webmining. Master thesis, Charles University, Prague, 2007 (In Czech).

Manning, Ch., D., Raghavan, P., and Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press, 2008.

Mecca, G., Raunich, S., and Pappalardo, A.: A new algorithm for clustering search results. Data & Knowledge Engineering. Volume 62, Issue 3 (September 2007) pp. 504-522, 2007

Maslowska, I.: Phrase-based hierarchical clustering of web search results. In: Advances in Information Retrieval – ECIR 2003, LNCS, vol.2633, Springer-Verlag Berlin, 2003, pp. 555-562

Osinski, O., Stefanowski, J., and Weiss, D.: Lingo: Search results clustering algorithm based on singular value decomposition. In: Proceedings of the International Conference on Intelligent Information Systems (IIPWM), 2004, pp. 369–377.

Procopiuc, C, Jones, M., Agarwal, P., and Murali. T.: A Monte Carlo algorithm for fast projective clustering. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, ACM Press, 2002, pp. 418–427.

Zamir O. and Etzioni, O.: Web document clustering: A feasibility demonstration. In: Proceedings of ACM SIGIR Conf. on Research and Development in Information Retrieval, 1998, pp. 46-54.