Vector model improvement by FCA and Topic Evolution

Jan Martinovič and Petr Gajdoš

Department of Computer Science, VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
{Petr.Gajdos, Jan.Martinovic}@vsb.cz

Abstract. Presented research is based on standard methods of information retrieval using the vector model for representation of documents (objects). The vector model is often expanded to get better precision and recall. In this article we have mentioned two approaches of vector model expansion. The first approach is based on hierarchical clustering. Its goal is to find a list of all documents they have most similar topic to the requested document. The second one is the document classification based on formal concept analysis. We have tried to evaluate all concepts and computed the importances of documents. At last have compared the results of our approach based on formal concept analysis and the results of classical vector model.

Keywords: Vector, FCA, Moebius, Topic Evolution, Clustering

1 Introduction

There are various systems for searching in document collections. They are based on vectors, probabilistic and another models for representation of documents, queries, rules and procedures. Each of the models contains a rank of limitations. Therefore we usually don't obtain all relevant documents. In our research we have to mentioned two approaches of vector model expansion. The first approach is based on hierarchical clustering. Its goal is to find a list of all documents they have most similar topics to the requested document. The second one is the document classification based on formal concept analysis. In following chapter, we have described classic vector model, cluster analysis and basic definition from formal concept analysis, which we needed for next computation. Then we have described our approaches for vector model improvement. In the fourth chapter, we have demonstrated benefits of our approach.

2 Background

2.1 Vector model

The vector model [12] of documents is dated back to 70th of the 20th century. In vector model there are documents and users queries represented by vectors.

We use m different terms $t_1 \dots t_m$ for indexing n documents. Then each document d_i is represented by a vector:

$$d_i = (w_{i1}, w_{i2}, \dots, w_{im}),$$

where w_{ij} is the weight of the term t_j in the document d_i.
An index file of the vector model is represented by matrix:

$$D = \begin{pmatrix} w_{11}\, w_{12} \dots w_{1m} \\ w_{21}\, w_{22} \dots w_{2m} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ w_{n1}\, w_{n2} \dots w_{nm} \end{pmatrix},$$

where i-th row matches i-th document, and j-th column matches j-th term.
In a vector model a query is represented by m dimensional vector:

$$q = (q_1, q_2, \dots, q_m),$$

where $q_j \in \langle 0, 1 \rangle^m$. On the basis of the query q we can compute *coefficient of similarity* for each document d_i. This coefficient can be understood as "distance" between the document's vector and the vector of the query. We used cosine measure for computing this similarity:

$$sim(q, d_i) = \frac{\sum_{k=1}^{m} (q_k w_{ik})}{\sqrt{\sum_{k=1}^{m} (q_k)^2 \sum_{k=1}^{m} (w_{ik})^2}}$$

The similarity of two documents is given by following formula:

$$sim(d_i, d_j) = \frac{\sum_{k=1}^{m} (w_{ik} w_{jk})}{\sqrt{\sum_{k=1}^{m} (w_{ik})^2 \sum_{k=1}^{m} (w_{jk})^2}}$$

For more information see [12].

2.2 Cluster analysis

The main goal of the cluster analysis is to find the fact, if there are any groups of similar objects. These groups are called *clusters*. We focuse on object clustering that can be divided in two steps. Firstly, we create the clusters and then we look for relevant clusters [7]. The reason of the cluster analysis is contained in the clusters hypothesis [12].

The searching process of an ideal fragmentation of objects is also called clustering. We use an agglomerative hierarchical clustering based on the similarity matrix:

At the beginning each object is considered as one cluster. Clusters are joined together in sequence. The algorithm is over, when all objects form only one cluster.

Similarity matrix Sim_C for collections C may be described with:

$$Sim_C = \begin{pmatrix} sim_{11}\, sim_{12} \ldots sim_{1n} \\ sim_{21}\, sim_{22} \ldots sim_{2n} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ sim_{n1} sim_{n2} \ldots sim_{nn} \end{pmatrix},$$

where i-th row matches i-th document and j-th column j-th document.

2.3 Formal Concept Analysis

FCA has been defined by R. Wille and it can be used for hierarchical order of objects based on object's features. The basic terms are formal context and formal concept. In this section there are all important definitions the one needs to know to understand the problematics.

Definition 1. *A formal context C = (G,M,I) consists of two sets G and M and a relation I between G and M. Elements of G are called objects and elements of M are called attributes of the context. In order to express that an object g is in a relation I with an attribute m, we write gIm or and read it as "the object g has the attribute m". The relation I is also called the incidence relation of the context.*

Definition 2. *For a set $A \subset G$ of objects we define*

$$A^{\uparrow} = \{m \in M \mid gIm \ for \ all \ g \in A\} \tag{1}$$

-the set of attributes common to the objects in A. Correspondingly, for a set $B \subset M$ of attributes we define

$$B^{\downarrow} = \{g \in G \mid gIm \ for \ all \ m \in B\} \tag{2}$$

-the set of objects which have all attributes in B.

Definition 3. *A formal concept of the context (G,M,I) is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A^{\uparrow} = B$ and $B^{\downarrow} = A$. We call A the extent and B the intent of the concept (A, B).*

Definition 4. *Let M be the totality of all features deemed relevant in the specific context, and let $I \subseteq G \times M$ be the incidence relation that describes the features possessed by objects, i.e. $(g, m) \in I$ whenever object $g \in G$ possesses a feature $m \in M$. For each relevant feature $m \in M$, let $\lambda(m) \geq 0$ quantify the importance or weight of feature m. The diversity value of a set S is defined as*

$$v(S) = \sum_{m \in M:(g,m) \in I \ for \ some \ g \in S} \lambda(m) \tag{3}$$

Our approach is also based on Conjugate Moebius Function and the on some properties go out from the Theory of diversity and Formal concept analysis.

Theorem 1. *For any function $v : 2^M \to \mathbb{R}$ with $v(\emptyset) = 0$ there exists unique function $\lambda : 2^M \to R$, the Conjugate Moebius Inverse function, such that $\lambda(\emptyset) = 0$ and for all S,*

$$v(S) = \sum_{A:A\cap S\neq\emptyset} \lambda(A) \tag{4}$$

Furthermore, the Conjugate Moebius Inverse λ is given by the following formula. For all $A \neq \emptyset$,

$$\lambda(A) = \sum_{A:A\cap S\neq\emptyset} (-1)^{|A|-|S|+1} * v(\overline{S}) \tag{5}$$

where $\overline{S}$ denotes the complement of S in M.

The diversity of an object (document) g is the sum of all weights of all features which are related to the object according to the incidence matrix. It conveys information about partial importance of an object but doesn't clearly display other dependences.

$$do(g) = \sum_{m:m\in M \ and \ (gIm)\in I} \lambda(m) \tag{6}$$

Next characteristic is called the sum of diversities of all objects. Actually, the objects of one concept can "cover" all features.

$$sdo(C) = \sum_{g:g\in C} do(g) \tag{7}$$

The importance of the object (document) g is the main point of our method. The value represents the importance from these aspects:

- Uniqueness - Is there any other similar object?
- Range of description - What type of dimension does the object describe?
- Weight of description - What is the weight of object in each dimension?

$$impo(g) = \sum_{C:C\ni g} \frac{sdo(C)}{v(S)} \lambda(A) \ do(g) \tag{8}$$

For more information see [3]

3 Vector Model Improvement

3.1 Using FCA to obtain the importance of documents

This method is based a) on the partial ordering of concepts in the concept lattice and b) on the inverse calculation of weights of objects using Moebius function and defined characteristics. Particular steps are illustrated by fig.[1] and briefly described in this chapter.

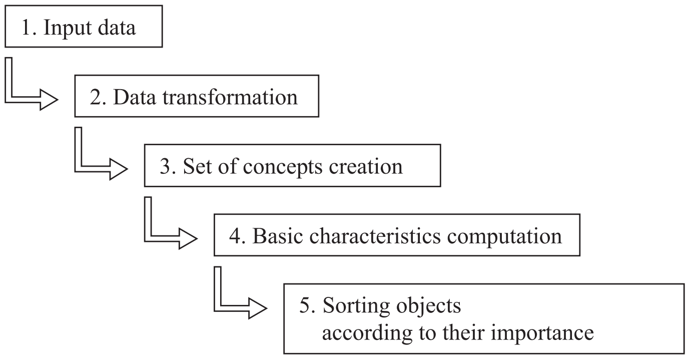

1. Input data
2. Data transformation
3. Set of concepts creation
4. Basic characteristics computation
5. Sorting objects according to their importance

Fig. 1. Getting importances of objects (documents)

First we obtain the input data (documents and words) like a table or matrix. The second step - scaling method is used to create an input incidence matrix. Every dimension can be scaled to a finite number of parts to get the binary values or we can only change non-zero values for number one, otherwise number zero. The output of transformation is an incidence matrix that we need as input for the concept calculation. Next the power set of concepts is computed using FCA algorithms. We can create the "concept lattice" and draw the Hasse diagram, but it's not important in our method. But it can be useful to show dependences between concepts, if we need it. We use only the list of concepts. After that, we can compute the basic characteristics for each concept according to the formulas (4), (5), (6), (7). Finally, we compute the importance of objects according to the formula (8). Obtained values provide us the criteria to sort the set of objects.

3.2 Evolution of topic

Our research concerns with the topics undergo an evolution. Lets assume document from collection of documents, that describes the same topic. It is clear, there are some other documents in the collection that describes the same topic, but they use different words to characterize the topic. The difference can be caused by many reasons. The first document focused on the topic use some set of words and next documents may use synonyms or for example exploration of new circumstances, new fact, new political situation etc. [4].

The result of searching an evolution of topic is to engaged query finding the lists of documents related by thematic with engaged query. We mean the query as query sets by terms or as document which is set as relevant.

We define this algorithm based on formal concept analysis and another algorithm for clustering. Our research gives us the answer for the question "What is the better way to improve the results of vector model?"

This is our algorithm using FCA and Moebius function:

Algorithm TOPIC-FCA:

1. We make the query transformation. It means that we create weighted vector of terms.
2. We compute the importances of documents (objects) by the formula 8. and we make the list of the documents and their importances.
3. We find the relevant document rel_d in the ordered list.
4. In finite steps, we look for "nearest" documents. The "nearest" document is the document, that has the smallest difference between its weight and the weight of rel_d. Founded document is excluded before next step.

Then we use this algorithm for clustering:

Algorithm TOPIC-CA:

1. We choose the total number of documents we want ('level').
2. We find leaf cluster which contain selected relevant document.
3. We get up in hierarchy.
4. We explore neighbouring clusters. First we select the cluster created on the highest sub-level. Each document, which we find, we add to the result list. When the count of all documents in the result list equal to 'level' we break finding.
5. We repeat the step 3.

3.3 Sort Response in Vector Model

The collection of documents responses to the query in the vector model, which is ordered by the coefficient of similarity of the query and the document. In this part, we present the method that can change this response by asking to the evolution of topic from clusters or concepts. Our approach is based on removing all non-relevant documents from the query and next on adding another relevant documents to the query. We have developed next algorithm for this change:

Algorithm SORT-EACH, this algorithm moves all documents in a result of the vector model query so that the documents belong to the same evolution of topic are closer to each other:

1. Collection of the documents from the vector query is marked as C_V.
2. The new sorted collection is marked as C_S and the count of its documents is a new value of the variable *count*.
3. We choose the total number of documents in evolution of topic and we mark it as *level*.
4. We do next sorting:

```
foreach document $D_V$ in $C_V$ do
    if $C_S$ is empty then
        add $D_V$ to collection $C_S$
        goto Continue
    end
    To document $D_V$ found by algorithm TOPIC-FCA (or TOPIC-CA)
        collection of evolution of topic $C_T$. Count of documents in topic is
        $level + 1$ (document $D_V$).
    foreach document $D_T$ in $C_T$ within document $D_V$ do
        if document $D_T$ is in $C_S$ then
            add the document $D_V$ behind $D_T$ do $C_S$
            goto Continue
        end
    end
    if not added $D_V$ then
        add $D_V$ to end of collection $C_S$
    label: Continue
end
```

5. We return collection C_S to user.

4 Illustrative sample of vector model improvement

Following tables show experimental results on generated data. Documents' importances were computed according to formula 8. The document selected by user is highlighted. This is the input document in TOPIC algorithms above. Each query is transformed to the vector of weights of terms. We use simplified matrix of documents and terms. The number "1" means that the document on the row consists the term in given column.

In the tables, we can see, that a vector queries give us worse results in some occurrence because they return zero-values of documents that don't have common terms. But, these documents can be about the same theme described by different terms (words). So we use the SEARCH-EACH algorithm for improve vector query by TOPIC-FCA or TOPIC-CA. We use the new TOPIC-FCA algorithm in these samples. See [4] to get another experiments.

Table 1. The results after inserted query "111111111111"

query	1	1	1	1	1	1	1	1	1	1	1	1		
	t_1	t_2	t_3	t_4	t_5	t_4	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	Document's importance	Vector query
doc. 1	1	1	1										36	0.5
doc. 2				**1**	**1**	**1**							**36**	**0.5**
doc. 3							1	1	1				36	0.5
doc. 4										1	1	1	36	0.5

Table 1 is very simple. We enter the query "111111111111". It means that we are looking for all relevant documents which contain all possible words. A vector query return all documents with the same relevancy because each of them contains three requested terms. Computed TOPIC-FCA (Importances of objects) brings zero improvement.

Table 2. The results after inserted query "111111111111"

query	1	1	1	1	1	1	1	1	1	1	1	1		
	t_1	t_2	t_3	t_4	t_5	t_4	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	Document's importance	Vector query
doc. 1	1	1	1		1								66.66666667	0.57735
doc. 2				**1**	**1**	**1**							**38**	**0.5**
doc. 3							1	1	1				36	0.5
doc. 4										1	1	1	36	0.5

Next, we describe table 2. The values of documents' importances show us the relative importances according to inserted query. There are only small differences between the importances of objects and vector query. The distance between document number 1 and selected document number 2 is larger then the distance between document number 2 and 3 (see the difference between documents' importances). The distance of the vector query and each document plus the distance between documents are the main reason of this appearance. It is better to describe this effect in the following table 3.

Table 3. The results after inserted query "000111000000"

query	0	0	0	1	1	1	0	0	0	0	0	0		
	t_1	t_2	t_3	t_4	t_5	t_4	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	Document's importance	Vector query
doc. 1	1	1	1	1	1	1	1	1	1	1	1	1	295	0.5
doc. 2				**1**	**1**	**1**							**37.33333**	**1**
doc. 3						1	1	1	1				54.4	0.288675
doc. 4										1	1	1	10.8	0

The vector query is "000111000000". We selected the document number 2 again. Although the first document contains the same terms as the second document, the distance between them is very large because of great number of terms the second document does not contain. Then, the evolution of topic of the second document is doc3, doc4 and at last doc1. So we get different ordering than the ordering after vector query.

The table 4 shows the main deficiency of the vector query. When we insert query "000111000000" we can not obtain the fourth document. But our method include this document because of a similarity to selected document number 2. So we can find new dependences between documents they can be about the same theme.

Table 4. The results after inserted query "000111000000"

query	0	0	0	1	1	1	0	0	0	0	0	0		
	t_1	t_2	t_3	t_4	t_5	t_4	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	Document's importance	Vector query
doc. 1				1	1			1	1	1	1	1	94.93333333	0.436436
doc. 2				**1**	**1**	**1**			**1**				**53.2**	**0.866025**
doc. 3						1	1	1	1				47	0.288675
doc. 4									1	1	1	1	26	0

Table 5. The results after inserted query "000111000000"

query	0	0	0	1	1	1	0	0	0	0	0	0		
	t_1	t_2	t_3	t_4	t_5	t_4	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	Document's importance	Vector query
doc. 1								1	1	1	1	1	41.86111111	0
doc. 2				**1**	**1**	**1**			**1**				**44.5**	**0.866025**
doc. 3						1	1	1	1				45.83333333	0.288675
doc. 4									1	1	1	1	28.6	0

The last table 5 shows better all hidden dependences between documents. The documents number 1 and 4 are not included in vector query, but we can say there can be some references between them because of common term number 9. The evolution of topic of selected document is doc3, doc1 and doc4.

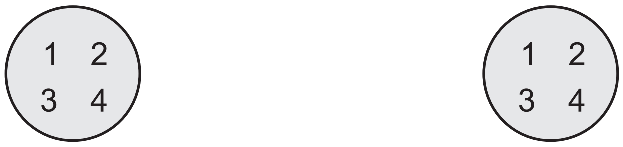
We tried to show the importance of our method in simple examples. If we use the TOPIC-FCA or TOPIC-CA for vector query improvement we can find another dependences between documents and we can get better ordering of requested documents.

4.1 Sample graphs

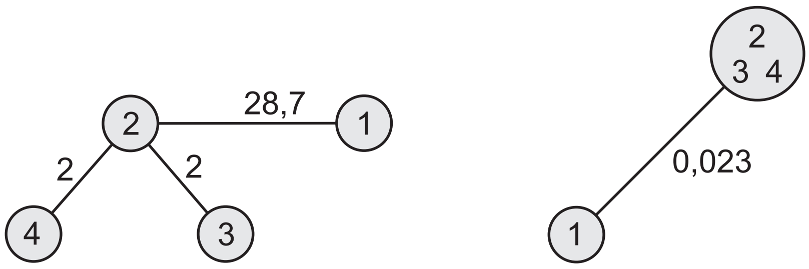
Following graphs show documents' distances from selected document number two. The graphs on the left show distances of documents after using TOPIC-FCA algorithm and the graphs on the right correspond to the results of the vector query. All distances were computed from selected document number 2.

Graph description:

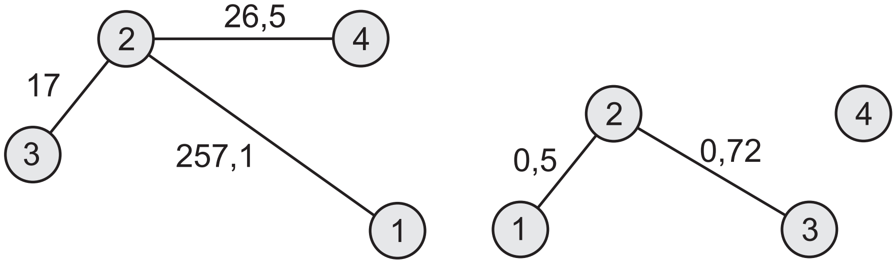
- Node represent a document or a cluster of document if the documents' distance is zero. Node's numbers correspond to number of documents.
- Edge connect comparable documents (nodes). The value means the distance of appropriate documents.



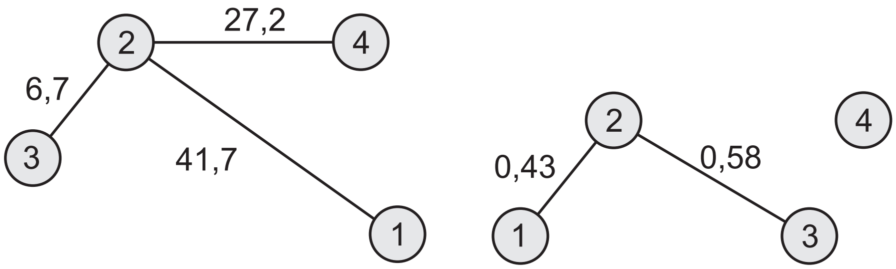
Documents' distances computed from table 1.



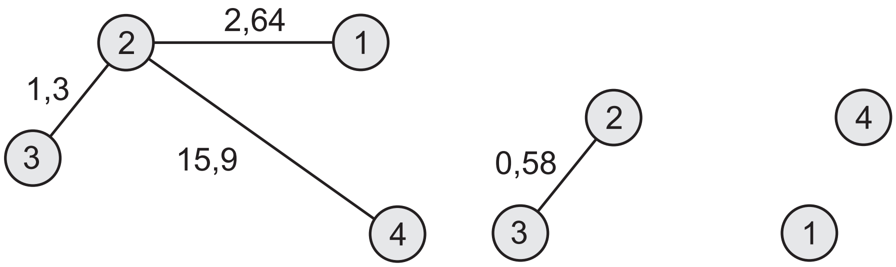
Documents' distances computed from table 2.



Documents' distances computed from table 3.



Documents' distances computed from table 4.



Documents' distances computed from table 5.

5 Conclusion and future work

We have described new method for vector query improvement based on formal concept analysis and Moebius inverse function. The known deficiencies of vector model have been suppressed using TOPICs and SEARCH-EACH algorithms. In the future work we would like to test our methods on real data. Our presented methods can be applied on small data sets or on large collections of documents.

References

1. Berry, M. W (Ed.): Survey of Text Mining: Clustering Classification, and Retrieval. Springer Verlag 2003.
2. Baeza-Yates R., Ribeiro-Neto B.: Modern Information Retrieval. Addison Wesley, New York, 1999.
3. Ďuráková, D., Gajdoš, P.: Indicators Valuation using FCA and Moebius Inversion Function. DATAKON, Brno, 2004, IBSN 80-210-3516-1

4. Dvorský J., Martinovič J., Snášel V.: Query Expansion and Evolution of Topic in Information Retrieval Systems, DATESO 2004, ISBN: 80-248-0457-3.
5. Dvorský J., Martinovič J., Pokorný J., Snášel V.: A Search topics in Collection of Documents.(in Czech),Znalosti 2004, ISBN: 80-248-0456-5.
6. Ganter B., Wille R.: Formal Concept Analysis. Springer-Verlag, Berlin, Heidelberg, 1999.
7. Christis Faloutsos, Douglas Oard: A Survey of Information Retrieval and Filtering Methods, Univ. of Maryland Institute for Advanced Computer Studies Report, College Park, 1995.
8. Keith Van Rijsbergen: The Geometry of Information Retrieva, Cambridge University Press, 2004.
9. Kummamuru K, Lotlikar R., Roy S., Singal K., Krishnapuram R.: A Hierarchical Monothetic Document Clustering Algorithm for Summarization and Browsing Search Results, WWW2004, New York, USA.
10. Nehring, K. and Puppe, C.: Modelling phylogenetic diversity. Resource and Energy Economics (2002).
11. Nehring, K.: A Theory of Diversity. Ecometrica 70 (2002) 1155 1198.
12. Pokorný J., Snášel V., Húsek D.: Dokumentografické informační systémy. Karolinum, Skriptum MFF UK Praha, 1998.
13. C.J. van Rijsbergen: Information Retrieval (second ed.). London, Butterworths, 1979.
14. Tsunenori Ishioka: Evaluation of Criteria for Information Retrieval, International Conference on Web Intelligence, IEEE Computer Society, 2003, ISBN 0-7695-1932-6.
15. S. Vempala: The Random Projection Method, Dimacs Series in Discrete Mathematics and Theoretical Computer Science, 2004.