Document Classification Using Itemsets

Jiří Hynek[*]
jiri.hynek@insite.cz

Karel Ježek[**]
jezek_ka@kiv.zcu.cz

Abstract: The essential point of this paper is to develop a method for automating time-consuming document classification in a digital library. The method proposed in this paper is based on itemsets, extending traditional application of the apriori algorithm.

Keywords: itemset, classification, class generation, cluster, clustering, apriori algorithm, document similarity, document categorization.

1 Introduction

1.1 Chapter Organization

Major difference between clustering and classification is described in the first part of this paper. The second part includes a description of the apriori algorithm applied to a real-world application. The novelty itemsets classification method is presented in the third part. Evaluation and motivation for further research are summarized in the fourth part.

1.2 Classification Versus Clustering

The objective of this paper is automatic classification of documents into specific topic areas. At this point we must emphasize the difference between two evidently distinct terms: *classification* and *clustering*. In case of *classification,* instances (documents) are associated with groups, or classes. Large volumes of data break apart into several discrete classes – these classes are a priori determined on the basis of a training data set. On the other hand, in case of *clustering* we are looking for groups of similar documents, maximizing intra-class similarity while minimizing inter-class similarity. Unlike in case of classification, we do not a priori know attributes characterizing each class. Therefore, at the end of clustering analysis, semantics must be assigned to each class by an analyst.

There are methods for document clustering by means of document-to-cluster similarity function (coefficient). Should the value of coefficient exceed threshold value θ, the document falls into the pertinent equivalence class.

Clustering can be based on a symmetric similarity matrix SIM with $k \times k$ dimension (having k documents in our library, $sim_{i,j}$ representing the document-to-document similarity function).

[*] inSITE, s.r.o., Rubešova 29, 301 53 Plzeň

[**] Department of Computer Science, the University of West Bohemia, Univerzitní 22, Plzeň

sim_{11} $\mathrm{sim}_{12}, \ldots, \mathrm{sim}_{1k}$

sim_{21} $\mathrm{sim}_{22}, \ldots, \mathrm{sim}_{2k}$

…

sim_{k1} $\mathrm{sim}_{k2}, \ldots, \mathrm{sim}_{kk}$

We can implement hierarchical clustering by setting different threshold values for each level of hierarchy. At the end we will obtain a *concept hierarchy* represented by a tree. We can use a graph, nodes represented by documents and edges drawn if $Sim\ (d_i, d_j) \geq \theta$. Cluster corresponds to a connected component in such a graph.

2 Apriori Algorithm

The Apriori algorithm, first proposed by Agrawal and Srikant is an efficient algorithm for knowledge mining in form of association rules [3]. We have recognized its convenience for document categorization. The original Apriori algorithm is applied to a transactional database of market baskets. In our case, instead of a market basket, we work with the basket of significant terms occurring in a text document and the transactional database is in fact a set of documents (represented by sets of significant terms). Consistently with the usual terminology let us denote terms as items and basket of terms (set of items) as an itemset.

Let π_i is an item, $\Pi = \{\pi_1, \pi_2, \ldots, \pi_m\}$ is an itemset and Δ is our database of documents. The itemset with k items is called k-itemset. Frequency of an itemset is defined as a simultaneous occurrence of items in the data being observed. Within our investigation we often utilize the threshold value θ employed for the minimum frequency of an itemset. Should frequency of an itemset exceed this threshold value, it is designated as a *frequent itemset* (we are trying to avoid the original – and rather confusing – term „large“). The transaction support in our case corresponds to the frequency of an itemset occurrence in the database Δ. Our goal is to discover frequent itemsets in order to characterize individual leaf topics in the digital library tree (topic tree).

Frequent itemsets' searching is an iterative process. At the beginning all frequent 1-itemsets are found, these are used to generate frequent 2-itemsets, then frequent 3-itemsets are found using frequent 2-itemsests, etc.

Let us suppose we have TD_S distinct significant terms in our text database Δ.

Firstly we generate candidates of frequent 1-itemsets (shortly candidate 1-itemsets). These are contained in our application directly in DF (Document Frequency) table. An example of candidate 1-itemsets and their frequencies is shown in tab.1:

itemset	frequency in %
$\{\pi_1\}$	**12**
$\{\pi_2\}$	0.5
$\{\pi_3\}$	**10**
$\{\pi_4\}$	3
$\{\pi_5\}$	**11**
$\{\pi_6\}$	**8**
. . .	. . .
$\{\pi_{TDS}\}$	2

Tab.1 Candidate 1-itemsets

Consequently, we compute frequent 1-itemsets. If predetermined minimal support θ shall be 7%[1], the resulting frequent 1-itemsets are boldfaced in tab.1.

In the next step, we generate 2-itemsets from frequent 1-itemsets. Suppose the result is in tab.2, where frequent 2-itemsets are boldfaced again.

itemset	frequency in %
$\{\pi_1, \pi_3\}$	**8**
$\{\pi_1, \pi_5\}$	**9.5**
$\{\pi_1, \pi_6\}$	**7**
$\{\pi_3, \pi_5\}$	**8.5**
$\{\pi_3, \pi_6\}$	4
$\{\pi_5, \pi_6\}$	5.5
. . .	. . .

Tab.2. Candidate 2-itemsets

Generation of subsequent candidate and frequent 3-itemsets continues as follows. If e.g. the itemset $\{\pi_1, \pi_3, \pi_5\}$ with frequency 7.1 is the only one recognized as frequent 3-itemset, the process of frequent itemsets' searching terminates with regard to Apriori property ("all non-empty subsets of a frequent itemset must be frequent"). While implementing this method, we utilize a technique similar to transaction reduction method: a document that does not contain a k-itemset can be left out of our further consideration, since it cannot contain any of (k+1)-itemsets.

3 Itemsets Classification Method

3.1 Terminology

Within the framework of this paper, we use the following notation:

Π = Frequent itemset

$|\Pi|$ = Cardinality of the itemset Π

T = Leaf topic (representing a categorization class)

D = Document

$\overline{D}$ (D bar) = A set of significant terms contained in document D

L = The number of leaf topics

N_i = The number of frequent itemsets having cardinality i

$D\Pi_i$ = The set of documents containing the itemset Π_i

DT_i = The set of documents associated with leaf topic T_i

$|DT_i|$ = The number of documents associated with leaf topic T_i

C_i = Set of itemsets characterizing leaf topic T_i

[1] The threshold value θ can vary for 1-itemsets, 2-itemsets, 3-itemsets, etc. Generally speaking, the higher the cardinality of an itemset, the lower value of θ can be chosen. Fine tuning of θ will be subject to further research.

On the basis of the apriori algorithm above, we will define frequent itemsets of various cardinalities.

For pairs: $\Pi_1, \Pi_2, \ldots, \Pi_{N2}$

For triplets: $\Pi_{N2+1}, \Pi_{N2+2}, \ldots, \Pi_{N3+N2}$

For quadruplets: $\Pi_{N3+N2+1}, \Pi_{N3+N2+2}, \ldots, \Pi_{N4+N3+N2}$

etc.

3.2 The Classification Problem

The task of classification in our information system is to serve two functions:

1. Document classification, i.e. automatic association of documents with topics (the principle can be used for knowledge classification in general);
2. Automatic identification of a context (topic) upon entering a user query.

The classification problem can be divided into two parts: *training phase* and *classification phase*. The training phase consists of the following:

- Defining a hierarchy (tree) of thematic areas (topics) by an expert;
- Manual insertion of a certain number of documents into topics (for the time being, more than 1200 documents), i.e. classification attributes are defined for each class (training data set). The expert defines L classification classes (L represents the number of leaf topics), categorizing all "training" documents available. Each topic should be assigned a statistically significant number of documents, say, at least ten.
- Automatic generation of representative itemsets of various cardinality for each topic (i.e. definition of clusters) – see below.

While performing classification, we utilize representative itemsets to classify documents into corresponding topics (each document is currently associated with 3.24 leaf topics on the average).

The classification algorithm can be evaluated in terms of accuracy and speed. Accuracy can be measured by means of a test-set, the members of which have a priori known classification. Accuracy is expressed as a ratio between the number of correct classifications and cardinality of the test-set. Complexity of such classification is determined by the number of topics (i.e. clusters) the classification algorithm must choose from.

3.3 Itemsets Classification Method

Training Phase

For each itemset Π we can find a characteristic set of documents containing Π. Let's designate this set of documents as $D\Pi$. It is obvious that cardinality of $D\Pi$ will be higher than a certain threshold value, since Π was selected as a frequent itemset.

Itemset Π_1 corresponds to the set $D\Pi_1$, Π_2 corresponds to $D\Pi_2$, etc. If we will work with pairs, triplets and quadruplets only, we will create N2 + N3 + N4 sets of documents. E.g. $D\Pi_i = \{D_2, D_{16}, D_{123}, \ldots, D_{765}\}$.

By analogy, for each leaf topic T_i there is a characteristic set of documents falling into this topic. Let's designate this set as DT_i. Topic T_1 corresponds to the set DT_1, topic T_2 to DT_2, etc. Altogether we will make L sets. For example, $DT_i = \{D_3, D_{16}, D_{125}, \ldots, D_{815}\}$.

Our goal is to specify a certain number of itemsets for each topic, where each itemset is associated with a subset of the set of leaf topics. Namely, itemset Π_j is associated with topic T_i corresponding to the values of $w_{\Pi j}$ exceeding some threshold value θ. The weight $w_{\Pi j}$ is calculated, for example, as follows:

$$w_{\Pi j} = \frac{\left|D\Pi_j \cap DT_i\right|}{\left|DT_i\right|} \qquad i = 1, 2, ..., L$$

$|DT_i|$ component is used for normalizing with the number of documents associated with the topic. Upon associating itemsets with individual topics based on the formula above, we will acquire sets of itemsets C representing particular topic T. For example, topic T_1 will correspond to $C_1 = \{\Pi_2, \Pi_5, \ldots, \Pi_{34}\}$, topic T_2 to $C_2 = \{\Pi_1, \Pi_6, \ldots, \Pi_{33}\}$ … etc. up to C_L.

Classification Phase

Within the process of document classification, we must take into account cardinality of itemsets in order to distinguish between correspondence in pairs and correspondence in quadruplets, for instance. That is why we define a weight factor corresponding to the cardinality of an itemset. For pairs we will use wf_2, for triplets wf_3, for quadruplets wf_4, etc. The higher the cardinality, the higher the weight factor.

Now we can proceed with classifying a document into a topic (or several topics). Let's suppose, set C_j contains elements $\Pi_1, \Pi_2, \ldots, \Pi_{|Cj|}$. We will compute the weight corresponding to the accuracy of associating document D with topic T_j:

$$W_{T_j} = \sum_{i=1}^{|C_j|} wf_{|\Pi_i|} \qquad where\ \Pi_i \in C_j \wedge \Pi_i \subseteq \overline{D}\ for\, all\ j = 1, 2, ..., L$$

In other words, the classification weight is determined by the sum of weight factors for all itemsets of a given topic, which (the itemsets) are contained in the document being classified. With some simplification, we could call the above criterion *the nearest neighbor criterion*.

The document will be associated with topic T_j corresponding to the highest weight $W_{Tj.}$ Naturally, we can desire to associate the document with several topics. If this is the case, we will classify the document to all topics T_j where W_{Tj} exceeds certain threshold value θ.

3.3 Automatic Identification of the Context

An analogy of the itemsets classification method can be used for another type of application: automatic identification of the context. The user may wish to enter his/her query at the root level of the tree of topics. Query at that moment can be treated as a document being classified using the itemsets classification method. The system returns a set of documents or topics corresponding to the query entered. This feedback can be used either as a systems response, or a semi-product that will be subject to further searching. We can as well expand the user query by terms specific for a given context and use this expanded query in further processing (such as full-text search). We believe that this approach could improve precision of the information retrieval process.

4 Evaluation and Further Research

The method proposed in this paper shows viable within the commercial application it will be applied to. The authors believe that by implementing the method the task of document classification will become more efficient.

Further research will focus on tuning the weigh-factor parameters wf_2, wf_3, etc. and observing their impact on the accuracy of classification. The study will also concentrate on the method of selecting itemsets and computing the weight $w_{\Pi j}$, as well as fine tuning the threshold value θ determining frequent itemsets.

The complexity of the Apriori algorithm is much dependent on selecting θ value for marking an itemset as frequent / non frequent. Implementation leads to a reasonable polynomial-bound problem. The complexity of the classification can be, with some approximation, expressed as follows:

$C_{AVG} \times L \times K$ (= $C_{TOT} \times K$), where C_{AVG} is the average number of itemsets in C, L is the number of leaf topics, K is a constant expressing complexity of matching itemsets in C with the document being classified[2] and C_{TOT} is the total number of itemsets over all classes C.

The complexity of classification can be further reduced by means of utilizing a modification of the apriori property. We can observe transitive dependence relationships among itemsets in class C and take advantage of this dependence: if an itemset Π is not contained in the document being classified (D), none of the itemsets in C dependent on Π can be contained in D. We can have, for example, the following dependencies:

$$\begin{tikzcd}
\{\pi 1, \pi 3\} \arrow[r] \arrow[d] & \{\pi 1, \pi 3, \pi 5\} & \{\pi 2, \pi 6\} \arrow[d] \\
\{\pi 1, \pi 3, \pi 6\} \arrow[dr] & & \{\pi 2, \pi 3, \pi 6\} \arrow[dl] \\
 & \{\pi 1, \pi 2, \pi 3, \pi 6\} &
\end{tikzcd}$$

Should this method, or its modification, prove efficient in the real-world environment, it will become a part of an extensive application – the digital technical library currently used by energy utilities within the Czech Republic.

References

1. *Bulletin of the Technical Committee on Data Engineering*, June 1998, Vol. 21, No. 2, IEEE Computer Society
2. Kotásek P., Zendulka J.: *AprioriItemset – A New Algorithm for Discovering Frequent Itemsets*, ISM 99, April 27-29, 1999, Czech Republic
3. Agrawal et al.: Advances in Knowledge Discovery and Data Mining, MIT Press 1996, pp. 307-328

[2] Constant K is clearly dependent on the size of the document being classified, however, it is important to note that doubling the number of terms in the document being classified does not necessarily mean doubling the classification time. We are using only significant terms (leaving stop words out), neglecting repetitive occurrence of these terms.