Frequent Itemset Mining for Clustering Near Duplicate Web Documents

Dmitry I. Ignatov and Sergei O. Kuznetsov

Higher School of Economics, Department of Applied Mathematics
Kirpichnaya 33/5, Moscow 105679, Russia
{skuznetsov,dignatov}@hse.ru

Abstract. A vast amount of documents in the Web have duplicates, which is a challenge for developing efficient methods that would compute clusters of similar documents. In this paper we use an approach based on computing (closed) sets of attributes having large support (large extent) as clusters of similar documents. The method is tested in a series of computer experiments on large public collections of web documents and compared to other established methods and software, such as biclustering, on same datasets. Practical efficiency of different algorithms for computing frequent closed sets of attributes is compared.

1 Introduction

Around 30% of documents in Internet have duplicates, which necessitates creation of efficient methods for computing clusters of duplicates [5,6,7,9,10,14,15,17,24,23]. The origin of duplicates can be different: from intended duplicating information on several severs by companies (legal mirrors) to cheating indexing programs of websites, illegal copying and almost identical spammer messages. Usually duplicates are defined in terms of similarity relation on pairs of documents: two documents are similar if a numerical measure of their similarity exceeds a certain threshold (e.g., see [5,6,7]). The situation is represented then by a graph where vertices are documents and edges correspond to pairs of the similarity relation. Clusters of similar documents are computed then as cliques or as connected components of such similarity graphs [7]. The main stages in finding clusters of duplicates are as follows (see, e.g., [7]): representing documents by sets of attributes, making concise document images by choosing subsets of images, defining similarity relation on document images, and computing clusters of similar documents. At the first stage of processing, after removing HTML-markup and punctuation marks, documents are turned into strings of words, which are, in turn, represented by sets of attributes. We have there two main different approaches, called syntactical and lexical. The syntactical approach consists in defining binary attributes that correspond to each fixed length substring of words (or characters). Such substrings are called shingles. Usually a shingle corresponds to a sequence of words and there are two parameters of shingling: the length of a shingle (the number of words in a shingle)

S. Rudolph, F. Dau, and S.O. Kuznetsov (Eds.): ICCS 2009, LNAI 5662, pp. 185–200, 2009.

and offset, the distance between the beginnings of two shingles. Each shingle is coded by a hash code (equal shingles have equal codes and it is unlikely that different shingles have same codes). Then, by means of a randomization scheme, a subset of shingles is chosen for a concise image of the document [5,6,7]. Such an approach is used in AltaVista search engine and (judging by patent [23]), in Google too. There are several principles for choosing number of shingles in an image: A fixed number, a logarithmic (as in Yandex mail service) number, linear number (each kth shingle), etc. In lexical methods representative words are chosen according to their significance of these words. Usually indices of significance are based on frequencies: those words whose frequencies lie in a certain interval (except for stop-words from a special list of about 30 stop-words with including articles, prepositions and pronouns) are taken: high frequency words can be noninformative and low frequency words can be either haphazard words that could have not appeared in a text or misprints.

In lexical methods such as I-Match [10] a large text corpus is used for generating lexicon, i.e., a set of representative words. A document is represented by the words that occur in the lexicon. In generation of the lexicon the words with lowest and highest frequencies are deleted, I-Match generates signature of a document (set of terms) and hash code of the document, where two documents get the same hash code with the probability equal to their similarity measure (according to the so-called cosine measure). As shown in [17], I-Match is sometimes instable to changes in texts, e.g., to *marginal* randomization of actually identical spammer messages. To avoid this drawback, besides one standard signature, one uses K more signatures, each of which is obtained by random deletion of certain amount of terms from the initial signature (i.e., all new signatures are subsets of the initial one). Two documents are considered to be almost duplicates if their images from K+1 signatures have *large* intersection in at least one of signature. In [17] the authors noted the similarity of this approach to the approach based on supershingles (concatenation of supershingles). In lexical method [15] the focus is towards the construction of a lexicon, a set of descriptive words, which should be concise, but cover well the collection. The occurrence of a word in a document image is robust with respect to small changes in the document. When document images are defined, one defines similarity relation on documents starting from a similarity measure which takes two documents to a number in the [0,1] interval depending on the amount of their common description units (shingles or words, in syntactical or lexical approaches, respectively). Then one chooses a threshold, exceeding which means large similarity of documents (the documents are near-duplicates). This metrics and a threshold define (symmetric and reflexive) similarity relation on document pairs. The similarity relation on document pairs determine clusters of near-duplicates. Definition of a cluster may also vary. A possible definition often used in practice, e.g., in Altavista search engine [7], is as follows: Consider the graph where Internet documents correspond to vertices and similarity relation corresponds to edges. Then a cluster of near duplicates is a connected component of such a graph. An advantage of such definition is efficiency of computation: a connected component may be computed in time

linear in the number of edges. An obvious drawback here is that the relation *to be near duplicates* is not transitive, therefore absolutely different documents can occur in a cluster. The strongest definition of a cluster arising from a similarity relation is that based on a graph clique. This definition is more adequate than that based on the connected component, but is much harder computationally, since generation of maximal cliques is a classical intractable problem. These two extreme definitions of a cluster admit for a broad scope of intermediate formulations that realize trade-off between precision and complexity of computing the clusters. Other methods of cluster definition are based on variations of standard methods of cluster analysis, e.g., when clustering a new object uses the distance to the *mass center* of clusters. Methods of this type essentially depend on the sequence in which objects to be clustered arrive. As applied to the problem of clustering duplicates, this means that documents that occurred earlier determine stronger the structure of cluster than documents that occurred later.

In this paper we consider similarity not as a relation on the set of documents, but as an operation taking each two documents to the set of all common elements of their concise descriptions. Here description elements are either syntactical units (*shingles*) or lexical units (*representative words*). A cluster of similar documents is defined as a set of all documents with a certain set of common description units. A cluster of duplicates is defined as a set of documents where the number of common description units exceeds a certain threshold. In this paper we compare results of its application (for various values of thresholds) with the list of duplicates obtained by applying other methods to the same collection of documents. We studied the impact of the following model parameters on the result:

- The use of the syntactical or lexical methods for representing documents,
- the use of method "*n minimal elements in a permutation*" or "*minimal elements in n permutations*"[5,6,7] (the second method, having better probability-theoretical properties, has worse computational complexity.)
- shingling parameter,
- threshold value of similarity of document images.

One of our goals was to relate computation of pairwise similarity with generation of clusters so that, on the one hand, the obtained clusters are independent of the order of document occurrence (in contrast to methods of cluster analysis) and, on the other hand, they would guarantee real pairwise similarity of all document images in the cluster. To this end we employed an approach based on formal concepts: Clusters of documents are given by formal concepts of the context where objects correspond to description units (units of a language describing documents: shingles, lexical units, etc.) and attributes are document names. A cluster of *very similar documents* corresponds then to a formal concept such that the size of extent (the number of common description units of documents) exceeds a threshold given by parameter. Thus, generating very similar documents is reduced to the problem of Data Mining [21] known as generating *frequent closed itemsets*.

The rest of the paper is organized as follows. In the second section we consider briefly a mathematical model underlying methods of composing document images and methods for finding clusters of similar documents. In the third section we give a short description of software tools and experiments with a large collection of web documents. In the fourth section we discuss results of computer experiment and set further problems.

2 Computational Model

2.1 Document Image

For creating document images we used standard syntactical and lexical approaches with different parameters. Within syntactical approach we realized the shingling scheme and computing document image (sketch) with the method "*n minimal elements in a permutation*" and the method "*minimal elements in n permutations*" detailed description of which can be found in [5,6,7]. For each text the program **shingle** with two parameters (*length* and *offset*) generate contiguous subsequences of size *length* such that the distance between the beginnings of two subsequent substrings is *offset*. The set of sequences obtained in this way is hashed so that each sequence receives its own hash code. From the set of hash codes that corresponds to the document a fixed size (given by parameter) subset is chosen by means of random permutations described in [5,6,7]. The probability of the fact that minimal elements in permutations on hash code sets of shingles of documents A and B (these sets are denoted by F_A and F_B, respectively) coincide, equals to the similarity measure of these documents $\mathrm{sim}(A, B)$:

$$sim(A, B) = P[min\{\pi(F_A)\} = min\{\pi(F_B)\}] = \frac{|F_A \cap F_B|}{|F_A \cup F_B|}$$

Permutations (that can be represented by renumbering of shingles) are realized by multiplying of binary vectors that represent document images (each component of such a vector corresponds to the hash code of a particular shingle from the image) on random binary matrices. For each hash code from the set of hash codes of a documents its number in each random permutation is computed as a product of the hash code given in the form of binary vector on the randomly generated binary matrix that corresponds to the permutation. The number of permutations is also a parameter. For each permutation (given by a binary matrix) the minimal element (i.e., hash code of a shingle that became the first after the permutation) is chosen. The image of a document in the method "*n minimal elements in a permutation*" is the set of n minimal (first) hash codes in a permutation. The image of a document in the method "*minimal elements in n permutations*" is the set consisting of minimal (first) hash codes in n independent permutations. In both methods the images of all documents have fixed length n. The second approach has better randomization properties (see [5,6,7] for details), however needs more time for computations (n times more than in the first approach).

2.2 Definition of Similarity and Similarity Clusters by Means of Frequent Concepts

First, we briefly recall the main definitions of Formal Concept Analysis (FCA) [11]. Let G and M be sets, called the set of objects and the set of attributes, respectively. Let I be a relation $I \subseteq G \times M$ between objects and attributes: for $g \in G,\ m \in M$, gIm holds iff the object g has the attribute m. The triple $K = (G, M, I)$ is called a *(formal) context*. Formal contexts are naturally given by cross tables, where a cross for a pair (g, m) means that this pair belongs to the relation I. If $A \subseteq G$, $B \subseteq M$ are arbitrary subsets, then *derivation operators* are given as follows:

$$A' := \{m \in M \mid gIm \text{ for all } g \in A\},$$
$$B' := \{g \in G \mid gIm \text{ for all } m \in B\}.$$

The pair (A, B), where $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$ is called a *(formal) concept (of the context K)* with *extent* A and *intent* B.

The operation $(\cdot)''$ is a closure operator, i.e., it is idempotent ($X'''' = X''$), extensive ($X \subseteq X''$), and monotone ($X \subseteq Y \Rightarrow X'' \subseteq Y''$). Sets $A \subseteq G$, $B \subseteq M$ are called *closed* if $A'' = A$ and $B'' = B$. Obviously, extents and intents are closed sets. Formal concepts of context are ordered as follows: $(A_1, B_1) \leq (A_2, B_2)$ iff $A_1 \subseteq A_2 (\Leftrightarrow B_1 \supseteq B_2)$. With respect to this order the set of all formal concepts of the context K makes a lattice, called a *concept lattice* $\underline{\mathfrak{B}}(K)$ [11].

Now we recall some definitions related to association rules in Data Mining. For $B \subseteq M$ the value $|B'| = |\{g \in G \mid \forall m \in B(gIm)\}|$ is called *support* of B and denoted by $\sup(B)$. It is easily seen that the set B is closed if and only if for any $D \supset B$ one has $\sup(D) < \sup(B)$. This property is used for the definition of a closed itemset in Data Mining. A set $B \in M$ is called *k-frequent* if $|B'| \leq k$ (i.e., the set of attributes B occurs in more than k objects), where k is parameter. Computing frequent closed sets of attributes (or itemsets) became important in Data Mining since these sets give the set of all association rules [21]. For our implementation where contexts are given by set G of description units (e.g., shingles), set M of documents and incidence (occurrence) relation I on them, we define a cluster of k-similar documents as intent B of a concept (A, B) where $|A| \geq k$. Although the set of all closed sets of attributes (intents) may be exponential with respect to the number of attributes, in practice contexts are *sparse* (i.e., the average number of attributes per object is fairly small). For such cases there are efficient algorithms for constructing all most frequent closed sets of attributes (see also survey [18] on algorithms for constructing all concepts). Recently, a competitions in time efficiency for such algorithms were organized in series of workshops on Frequent Itemset Mining Implementations (FIMI). By now, a leader in time efficiency is the algorithm FPmax* [13]. We used this algorithm for finding similarities of documents and generating clusters of *very similar documents*. As mentioned before, objects are description units (shingles or words) and attributes are documents. For representation of this type *frequent closed itemsets* are closed sets of documents, for which the number

of common description units in document images exceeds a given threshold. Actually, FPmax* generates *frequent itemsets* (which are not necessarily closed) and *maximal frequent itemsets*, i.e., frequent itemsets that are maximal by set inclusion. Obviously, maximal frequent sets of attributes are closed.

3 Program Implementation

Software for experiments with syntactical representation comprise the units that perform the following operations:

1. XML Parser (provided by Yandex): it parses XML packed collections of web documents
2. Removing html-markup of the documents
3. Generating shingles with given parameters length-of-shingle, offset
4. Hashing shingles
5. Composition of document image by selecting subsets (of hash codes) of shingles by means of methods *n minimal elements in a permutation* and *minimal elements in n permutations*.
6. Composition of the inverted table the list of identifiers of documents shingle thus preparing data to the format of programs for computing closed itemsets.
7. Computation of clusters of *k-similar documents* with FPmax* algorithm: the output consists of strings, where the first elements are names (ids) of documents and the last element is the number of common shingles for these documents.
8. Comparing results with the existing list of duplicates (in our experiments with the ROMIP collection of web documents, we were supplied by a precomputed list of duplicate pairs).

This unit outputs five values: 1) the number of duplicate pairs in the ROMIP collection, 2) the number of duplicate pairs for our realization, 3) the number of unique duplicate pairs in the ROMIP collection, 4) the number of unique duplicate pairs in our results, 5) the number of common pairs for the ROMIP collection and our results. For the lexical method, the description units are words (not occurring in the stop list) the frequencies of which lie in a certain interval. The amount of words in the dictionary is controlled by making closer the extreme points of the interval.

4 Experiments with ROMIP Data

As experimental data we used ROMIP collection of URLs (see www.romip.ru) consisting of 52 files of general size 4.04 GB. These files contained 530 000 web pages from narod.ru domain. Each document from the collection has size greater or equal to 10 words. For experiments the collection was partitioned into several parts consisting of three to 24 such files (from 5% to 50% percent of the whole collection). Shingling parameters used in experiments were as follows: the

number of words in shingles was 10 and 20, the offset was always taken to be 1 (which means that the initial set of shingles contained all possible contiguous word sequences of a given length). Two methods of composing document image described in Section 2.1 were studied: *n minimal elements in a permutation* and *minimal elements in n permutations*. The sizes of resulting document images were taken in the interval 100 to 200 shingles. In case of lexical representation described in Section 2.1, only words from the resulting dictionary were taken in the document image (the set of descriptive words). As frequency thresholds defining *frequent closed sets* (i.e., the numbers of common shingles in document images from one cluster) we experimentally studied different values in intervals, where the maximal value is equal to the number of shingles in the document image, e.g., [85, 100] for document images with 100 shingles, the interval [135, 150] for document images of size 150, etc. Obviously, choosing the maximal value of an interval, we obtain clusters where document images coincide completely.

For parameters taking values in these intervals we studied the relation between resulting clusters of duplicates and ROMIP collection of duplicates, which consists of pairs of web documents that are considered to be near duplicates. Similarity of each pair of documents in this list is based on Edit Distance measure, two documents were taken to be duplicates by authors of this testbed if the value of the Edit Distance measure exceeds threshold 0.85 [27]. As we show below, this definition of a duplicate is prone to errors, however making a testbed by manual marking duplication in a large web document collection is hardly feasible. Unfortunately, standard lists of near-duplicates are missing even for standard corpora such as TREC or Reuters collection [22]. For validating their methods, researchers create ad-hoc lists of duplicates using slightly transformed documents from standard collections.

In our study for each such pair we sought an intent that contains both elements of the pair, and vice versa, for each cluster of *very similar documents* (i.e., for each corresponding closed set of documents with more than k common description units) we take each pair of documents in the cluster and looked for the corresponding pair in the ROMIP collection. As result we obtain the table with the number of common number of near duplicate pairs found by our method (denoted by HSE) and those in the ROMIP collection, and the number of unique pairs of HSE duplicates (document pairs occurring in a cluster of "very similar documents" and not occurring in the ROMIP collection). The results of our experiments showed that the ROMIP collection of duplicates, considered to be a benchmark, is far from being perfect. First, we detected that a large number of false duplicate pairs in this collection due to similar framing of documents. For example the pages with the following information in table 1 about historical personalities 1 and 2 were declared to be near duplicates.

However these pages, as well as many other analogous false duplicate pairs in ROMIP collection do not belong to concept-based (maximal frequent) clusters generated in our approach.

In our study we also looked for *false duplicate clusters* in the ROMIP collection, caused by transitive closure of the binary relation "X is a duplicate of Y"

Table 1. Information about historical personalities

1. Garibald II, Duke of Bavaria	2. **Giovanni, Duke of Milan**
Short information:	Short information:
Full Name: Garibald	Full Name: Giovanni Visconti
Date of birth: unknown	Date of birth: unknown
Place of birth: unknown	Place of birth: unknown
Date of death: 610	Date of death: 1354
Place of death: unknown	Place of death: unknown
Father: Tassilo I Duke of Bavaria	Father: Visconti Matteo I, the Great Lord of Milan
Mother: uknown	Mother: uknown

(as in the typical definition of a document cluster in [7]). Since the similarity relation is generally not transitive, the clusters formed by transitive closure of the relation may contain absolutely nonsimilar documents. Note that if clusters are defined via maximal frequent itemsets (subsets of attributes) there cannot be effects like this, because documents in these clusters share necessarily large itemsets (common subsets of attributes).

We analyzed about 10000 duplicate document pairs and found four *false duplicate clusters*. An example is a cluster of 58 documents containing the webpages

aadobr.narod.ru/Foto/fotofr.html
avut.narod.ru/pages/page02.htm
azer.narod.ru/index.html
b-tour.narod.ru/index.html
barents.narod.ru/foto_nov.html
...

There is no cluster like this generated in our approach. Instead, we have several clusters of similar documents that have a certain amount (depending on the parameter) of common features.

4.1 Performance of Algorithms and Their Comparison

We measured time elapsed on the stages of shingling, composing document images and generating clusters of similar documents (by algorithms for computing frequent maximal itemsets). On the last stage we used and compared various algorithms: several well-known algorithms from Data Mining [12] and AddIntent, an algorithm which proved to be one of the most efficient algorithms for constructing the set of all formal concept and concept lattices [20].

Experiments were carried out on a PC P-IV HT with 3.0 MHz frequency, 1024 MB RAM under Windows XP Professional. Experimental results and time elapsed are partially represented in Tables 2-6.

Results of the method *"n minimal elements in a permutation"***.** For the dataset narod.1-6.xml, which contained 53 539 documents, the following shingling parameters were taken: size of document image 150 shingles, length of shingle 20, offset 1 character.

Table 2. Results of the method "*n minimal elements in a permutation*"

FPmax		All Pairs of Duplicates		Unique pairs of duplicates		Common pairs
Input	Threshold	ROMIP	HSE	ROMIP	HSE	
b_1_20_s_100_n1-12.txt	100	105570	15072	97055	6557	8515
b_1_20_s_100_n1-12.txt	95	105570	20434	93982	8846	11588
b_1_20_s_100_n1-12.txt	90	105570	30858	87863	13151	17707
b_1_20_s_100_n1-12.txt	85	105570	41158	83150	18738	22420
b_1_20_s_100_n1-24.txt	100	191834	41938	175876	25980	15958
b_1_20_s_100_n1-24.txt	95	191834	55643	169024	32833	22810
b_1_20_s_100_n1-24.txt	90	191834	84012	155138	47316	36696
b_1_20_s_100_n1-24.txt	85	191834	113100	136534	57800	55300
b_1_10_s_150_n1-6.txt	150	33267	6905	28813	2451	4454
b_1_10_s_150_n1-6.txt	145	33267	9543	27153	3429	6114
b_1_10_s_150_n1-6.txt	140	33267	13827	24579	5139	8688
b_1_10_s_150_n1-6.txt	135	33267	17958	21744	6435	11523
b_1_10_s_150_n1-6.txt	130	33267	21384	19927	8044	13340
b_1_10_s_150_n1-6.txt	125	33267	24490	19236	10459	14031

Table 3.

Time elapsed, s	Precision	Recall	Threshold	F1
1,2	0,4	0,2	100	0,24
2,0	0,4	0,2	95	0,31
3,1	0,5	0,4	90	0,42
5,3	0,5	0,4	85	0,44
7,2	0,4	0,5	80	0,46

For evaluating we used a popular measure that combines Precision and Recall: the weighted harmonic mean of precision and recall, or F-measure $= 2 \cdot (\text{precision} \cdot \text{recall})/(\text{precision} + \text{recall})$. This is also known as the $F1$-measure, because recall and precision are evenly weighted. The results obtained are given in Table 3, Fig. 1 and Fig. 2.

For the dataset narod.1.xml, which contained 6941 documents, the following shingling parameters were taken: size of document image 100 shingles, length of shingle 10, offset 1 character. The results were obtained are given in Fig. 3 and Table 4.

Comparing results of FPmax with results obtained with Cluto. In our experiments with Cluto we chose the repeated-bisecting algorithm that uses the cosine similarity function with a 10-way partitioning (ClusterRB), which is mostly scalable according to its author [16]. The number of clusters is a parameter, documents are given by sets of attributes, fingerprints in our case. The algorithm outputs a set of disjoint clusters. Algorithms from FIMI repository can process very large datasets, however, to compare with Cluto (which is much more time consuming as we show below) we took collection narod.1.xml that contains 6941 documents.

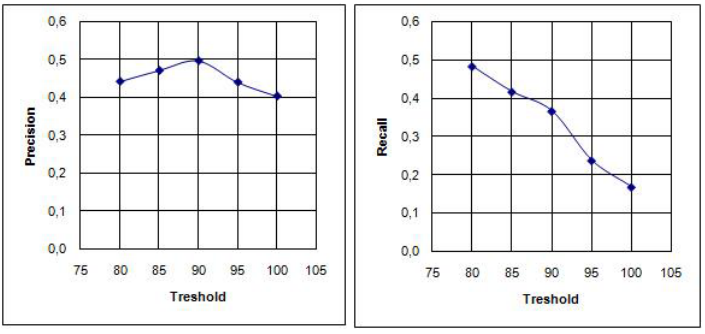

Fig. 1. Results of FPmax* on narod1-6.xml collection

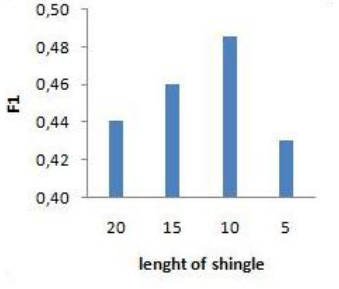

Fig. 2. F-measure vs shingle length

Table 4.

Time elapsed,s	Precision	Recall	Threshold	F1
0,098	0,76	0,25	150	0,38
0,128	0,74	0,29	145	0,42
0,187	0,70	0,39	140	0,50
0,276	0,67	0,50	135	0,57
0,383	0,63	0,57	130	0,60
0,455	0,58	0,64	125	0,61
0,559	0,47	0,64	120	0,54
0,669	0,37	0,67	115	0,48
0,873	0,29	0,70	110	0,41
1,045	0,23	0,73	105	0,35
1,294	0,18	0,69	100	0,29

Graphs and tables show that for 5000 clusters the output of ClusterRB has almost the same value of F-measure (0.63) as FPmax* for threshold 150 (F1=0,61). However, computations took 4 hours for ClusterRB and half a second for FPmax*.

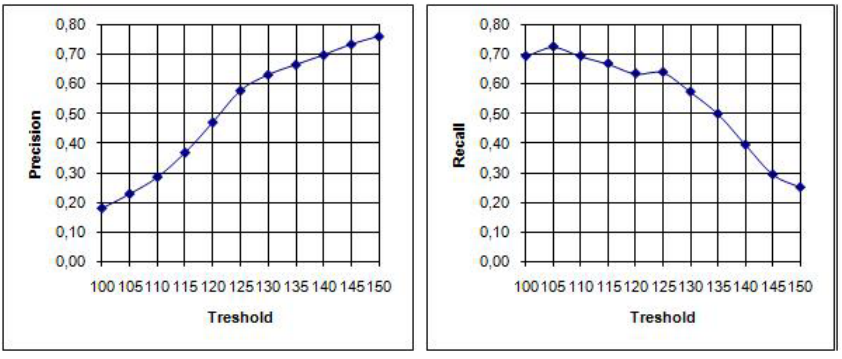

Fig. 3. Results of FPmax* on narod1.xml collection

Table 5.

Time, s	Precision	Recall	Number of clusters	F1
11	0,02	0,90	100	0,04
766	0,09	0,78	1000	0,16
3125	0,19	0,74	2000	0,30
6402	0,28	0,71	3000	0,40
14484	0,64	0,61	5000	0,63
19127	0,90	0,35	6000	0,51

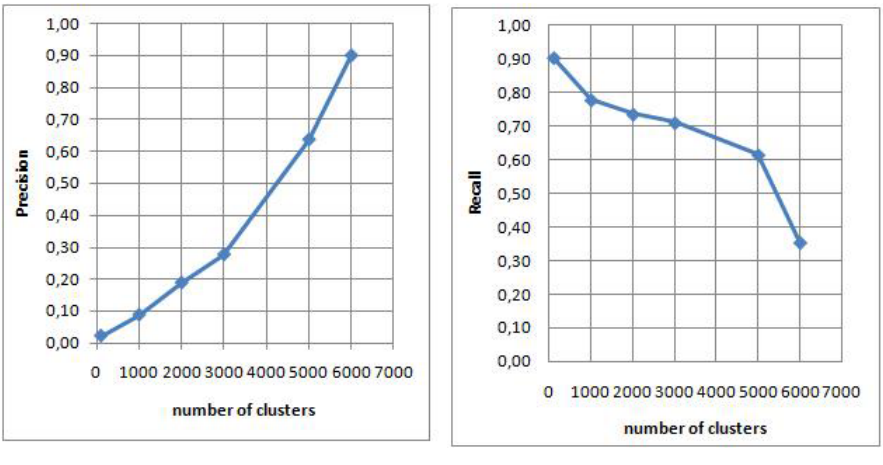

Fig. 4. Results of Cluto on narod1.xml collection

For same data collection narod.1.xml we made comparison to D-miner algorithm [3] and biclustering algorithms from BicAT system [2]. D-miner takes input in FIMI format, but computations were not completed due to lack of memory. Same effect was observed for biclustering algorithms from BicAT, which cannot

Table 6. Results for the method "*minimal elements in n permutations*"

FPmax		All Pairs of Duplicates		Unique pairs of duplicates		Common pairs
Input	Threshold	ROMIP	HSE	ROMIP	HSE	
m_1_20_s_100_n1-3.txt	100	16666	4409	14616	2359	2050
m_1_20_s_100_n1-3.txt	95	16666	5764	13887	2985	2779
m_1_20_s_100_n1-3.txt	90	16666	7601	12790	3725	3876
m_1_20_s_100_n1-3.txt	85	16666	9802	11763	4899	4903
m_1_20_s_100_n1-6.txt	100	33267	13266	28089	8088	5178
m_1_20_s_100_n1-6.txt	95	33267	15439	26802	8974	6465
m_1_20_s_100_n1-6.txt	90	33267	19393	24216	10342	9051
m_1_20_s_100_n1-12.txt	100	105570	21866	95223	11519	10347
m_1_20_s_100_n1-12.txt	95	105570	25457	93000	12887	12570

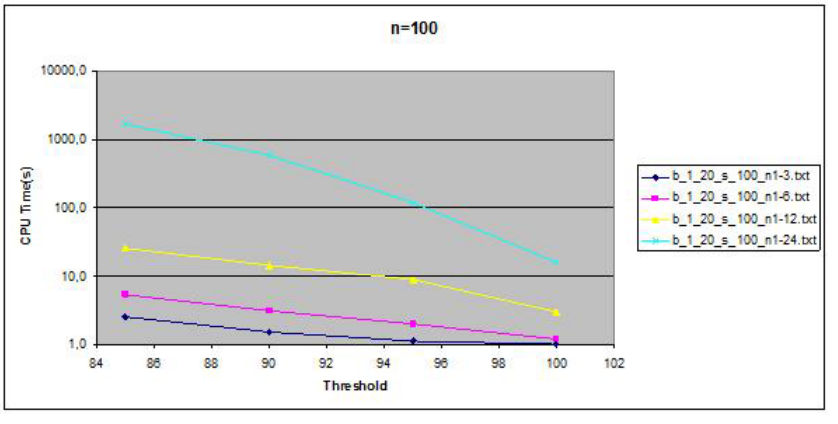

Fig. 5. Time spent by FPmax* for the method "n minimal elements in a permutation"

work with condensed representation and required inputting datatable of size 1.9 Gb in case of our document collection narod.1.xml.

FPmax* algorithm for the method *"minimal elements in n permutations"*. For large collections of documents and lower thresholds FPmax* did not complete computation due to insufficient memory.

In our experiments the best performance is attained by FPmax* algorithm, followed by the AFOPT algorithm [19]. These two algorithms proved to be the fastest in FIMI competitions [12]. AddIntent* (AddIntent modified for maximal frequent itemsets) lags behind these two, however, performs much better than MAFIA [8]. Optimized implementations of APRIORI and ECLAT [4] failed to compute the output even in the case of small subcollections of documents (about 4000 documents or 1% of the whole collection). These relative behaviour of algorithms is similar to that observed in [12] in experiments with low support. In the following Table we present running times in a typical experiment with different algorithms on a subcollection of about 10% of the whole collection.

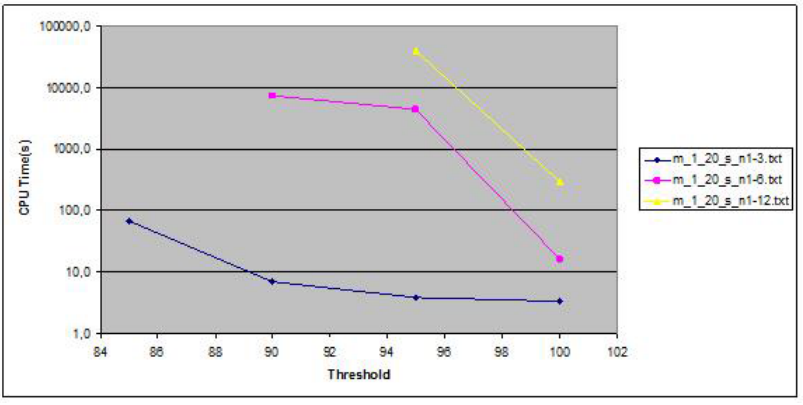

Fig. 6. Time spent by FPmax* for the method "minimal elements in n permutations"

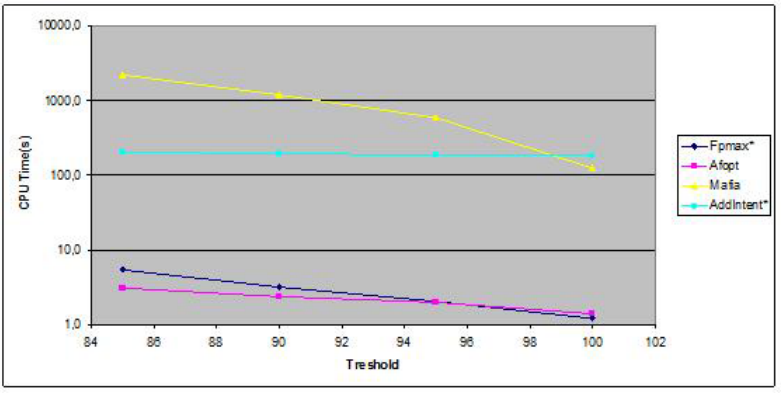

Fig. 7. Comparing efficiency of algorithms that compute maximal closed sets

In the contexts corresponding to these subcollections the number of objects is relatively large as compared to the minsup threshold value defined by parameters in the definition of duplicates. Thus, these are typical problems of generating frequent itemsets in low-support data and relative performance of data mining algorithms in our experiments is similar to that in survey [12].

5 Conclusion and Future Research

Analyzing results of our experiments with concept-based definition of clusters of similar documents with ROMIP data collection we can draw the following conclusions:

– The approach proposed in this paper allows for detecting false duplicates, as shown in experiments with ROMIP near-duplicate collection.

- Approaches based on closed sets of attributes propose adequate and efficient techniques for both determining similarity of document images and generating clusters of very similar documents. They can be efficiently used on the stage of outputting documents relevant to a query, when the number of all found relevant documents does not exceed several thousands (around 10000 documents). However, these algorithms may encounter major difficulties in treating larger collections of documents due to intrinsic exponential worst-case complexity of the problem of computing maximal frequent itemsets.
- For our datasets (which are very "column-sparse") the best Data Mining algorithms for computing frequent closed itemsets, FPmax* and Afopt, outperform AddIntent, one of the best algorithm for constructing concept lattice, adapted for computing maximal frequent itemset.
- Results of syntactical methods essentially depend on the parameter *shingle length*. Thus, in our experiments, for the shingle length 10 the results (pairs of duplicates) were much closer to those in the ROMIP collection as for the lengths of shingles equal to 20, 15, and 5.
- In our experiments the results obtained by different methods of document representation – *n minimal elements in a permutation* and *minimal elements in n permutations* – did not differ much, which testifies in favor of the first, faster method.
- Further work on generating clusters of web duplicates as formal concepts will be related to efficiency issues: more efficient algorithms and new definitions of suitable subsets of all clusters covering a collection of documents.

Acknowledgments

This research was supported by Russian Foundation for Basic Research, project 08-07-92497-NTsNIL_a. The authors would like to thank Sergei A. Obiedkov and Mikhail V. Samokhin for helpful discussions and participating in software realization of the approach.

References

1. Hasnah, A.M.: A New Filtering Algorithm for Duplicate Document Based on Concept Analysis. Journal of Computer Science 2(5), 434–440 (2006)
2. Barkow, S., Bleuler, S., Prelic, A., Zimmermann, P., Zitzler, E.: BicAT: a biclustering analysis toolbox. Bioinformatics 22(10), 1282–1283 (2006)
3. Besson, J., Robardet, C., Boulicaut, J.-F.: Constraint-based mining of formal concepts in transactional data. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS, vol. 3056, pp. 615–624. Springer, Heidelberg (2004)
4. Borgelt, C.: Efficient Implementations of Apriori and Eclat. In: Proc. Workshop on Frequent Itemset Mining Implementations Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2003) (2003)

5. Broder, A.: On the resemblance and containment of documents. In: Proc. Compression and Complexity of Sequences (SEQS: Sequences 1997)
6. Broder, A., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-Wise Independent Permutations. In: Proc. STOC, pp. 327–336 (1998)
7. Broder, A.: Identifying and filtering near-duplicate documents. In: Giancarlo, R., Sankoff, D. (eds.) CPM 2000. LNCS, vol. 1848, pp. 1–10. Springer, Heidelberg (2000)
8. Burdick, D., et al.: MAFIA: A Performance Study of mining Maximal Frequent Itemsets. In: Proc. Workshop on Frequent Itemset Mining Implementations Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2003) (2003)
9. Cho, J., Shivakumar, N., Garcia-Molina, H.: Finding replicated web collections. In: Proc. SIGMOD Conference, pp. 355–366 (2000)
10. Chowdhury, A., Frieder, O., Grossman, D.A., McCabe, M.C.: Collection statistics for fast duplicate document detection. ACM Transactions on Information Systems 20(2), 171–191 (2002)
11. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
12. Goethals, B., Zaki, M.: Advances in Frequent Itemset Mining Implementations: Introduction to FIMI 2003. In: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, FIMI 2003 (2003)
13. Grahne, G., Zhu, J.: Efficiently Using Prefix-trees in Mining Frequent Itemsets. In: Proc. FIMI 2003 Workshop (2003)
14. Haveliwala, T.H., Gionis, A., Klein, D., Indyk, P.: Evaluating Strategies for Similarity Search on the Web. In: Proc. WWW 2002, Honolulu, pp. 432–442 (2002)
15. Ilyinsky, S., Kuzmin, M., Melkov, A., Segalovich, I.: An efficient method to detect duplicates of Web documents with the use of inverted index. In: Proc. of the 11th International World Wide Web Conference, WWW 2002, Honolulu, Hawaii, USA, May 7-11. ACM, New York (2002)
16. Cluto, G.K.: A Clustering Toolkit. University of Minnesota, Department of Computer Science Minneapolis, MN 55455, Technical Report: 02-017, November 28 (2003)
17. Kolcz, A., Chowdhury, A., Alspector, J.: Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization. In: Kim, W., Kohavi, R., Gehrke, J., DuMouchel, W. (eds.) Proc. KDD 2004, Seattle, pp. 605–610 (2004)
18. Kuznetsov, S.O., Obiedkov, S.A.: Comparing Performance of Algorithms for Generating Concept Lattices. Journal of Experimental and Theoretical Artificial Intelligence 14, 189–216 (2002)
19. Liu, G., Lu, H., Yu, J.X., Wei, W., Xiao, X.: AFOPT: An Efficient Implementation of Pattern Growth Approach. In: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2003) (2003)
20. van der Merwe, D., Obiedkov, S.A., Kourie, D.: AddIntent: A New Incremental Algorithm for Constructing Concept Lattices. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 372–385. Springer, Heidelberg (2004)
21. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. Inform. Syst. 24(1), 25–46 (1999)
22. Potthast, M., Stein, B.: New Issues in Near-duplicate Detection, in Data Analysis. In: Machine Learning and Applications. Springer, Heidelberg (2007)

23. Pugh, W., Henzinger, M.: Detecting duplicate and near-duplicate files, United States Patent 6658423 (December 2, 2003)
24. Shivakumar, N., Garcia-Molina, H.: Finding near-replicas of documents on the web. In: Atzeni, P., Mendelzon, A.O., Mecca, G. (eds.) WebDB 1998. LNCS, vol. 1590, pp. 204–212. Springer, Heidelberg (1999)
25. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW 2008: Proceeding of the 17th international conference on World Wide Web, Beijing, China, pp. 131–140 (2008)
26. Zhao, Y., Karypis, G.: Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering. Machine Learning 55, 311–331 (2004)
27. `http://company.yandex.ru/academic/grant/datasets_description.xml`