FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 19, no. 3, December 2006, 439-451

Text Classification: Forming Candidate Key-Phrases from Existing Shorter Ones

Nikitas N. Karanikolas and Christos Skourlas

Abstract: The hard problem of the Text Classification usually has various aspects and potential solutions. In this paper, two main research directions for narrative documents' classification are considered. The first one is based on data mining and rule induction techniques, while the second combines the traditional Text Retrieval techniques (use of the vector space model, index terms, and similarity measures), Natural Language Processing and Instance based Learning techniques. Key-phrases can be used as attributes for mining rules or as a basis for measuring the similarity of new (unclassified) documents with existing (classified) ones. Hence, we eventually focus on the problem of extracting key-phrases from text's collection in order to use them as attributes for text classification. A new algorithm for the discovery of key-phrases is described. Candidate key-phrases are built using frequent smaller ones and special emphasis is given to the reduction of the complexity of the algorithm.

Keywords: Text classification, key-phrase extraction, text indexing, information retrieval, document management.

1 Introduction

Text Classification could be defined as the application of (semi) automatic methods in order to choose, from a set of predefined classification codes, the appropriate one (category / class) for a given new document. As an example, patient discharge letters could be semi-automatically classified using some technique based on the selection of the appropriate ICD (International Classification of Deceases and Diagnoses) code [1].

Manuscript received May 20, 2006.

The authors are with Technological Educational Institute of Athens (TEI-A), Department of Informatics, Ag. Spyridonos Street, 12210 Athens, Greece (e-mails: nnk@teiath.gr and cskourlas@hol.gr).

Various studies have focused on the construction of a model (Rule or Tree) related to the existence of key-phrases in order to assign the class of the unclassified document. Such a method usually uses training sets of documents, already classified, and a predefined list of key-phrases. Consequently, it creates a vector, for each document of the training set, that represents the existence or not of the predefined key-phrases in the document. The last item of each vector is the class code (the label) of document. The following figure depicts the vector and its relationship with the list of key-phrases and the list of available classes.

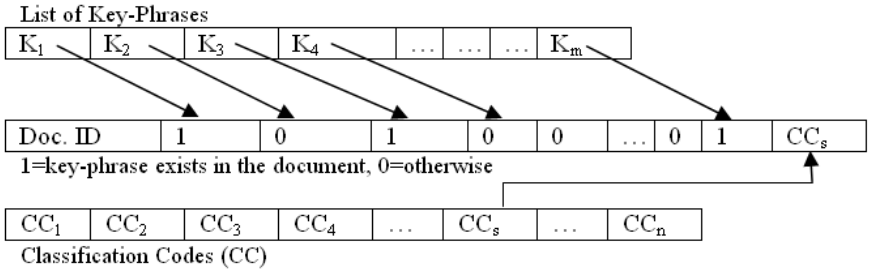

Fig. 1. The representation of classified documents.

The labeled vectors of the documents form a table of m+1 attributes where Data Mining algorithms could be applied to construct classification Rules. For example, the lists of the representative key-phrases (one such list for each class of documents) are used for extracting naive rules [2], that calculate the similarity of a new document with each class CL_i. Similarity is calculated as the number of the items of the (CL_i class' representative key-phrases) list that actually exist in the new document, divided (normalized) by the population of the list.

The following notation denotes the list of representative key-phrases of class CL_i:

$$RCL_i = \{kpCL_{i1}, kpCL_{i2}, ..., kpCL_{ir}\}$$

where $kpCL_{ij}$ is the j-(key-phrase) of the representative key-phrases of class CL_i. The measure $|RCL_i|$ defines the number of the representative key-phrases for the specified class CL_i.

A simple framework for extracting naive rules that measure the similarity of a new, unclassified, document with the class CL_i is defined by the equation:

$$S'(D_{new}, CL_i) = \frac{count(exist(kpCL_{ij}, D_{new}))}{|RCL_i|}$$

where the operation (function) "exists" denotes if a representative key-phrase $kpCL_{ij}$ exists in the new document and "counts" means the calculation of the existing key-phrases. An interesting characteristic of the Authority list, which is constructed by

the algorithm in section 2, is that it contains key-phrases that are frequent within the documents of only one or few classes. Consequently, for each class we can have a list of ”representative” key-phrases.

Another approach for the classification of documents is based on the similarity between existing documents (the ”training” set) and the new (unclassified) documents (e.g. [3]). Such an Instance based learning method assumes that similar documents must be classified in the same category (class) or in other words must share the same classification code. In such a method the list of key-phrases is a promising set of attributes that can be used to describe and discriminate between existing and new documents. This approach exploits the knowledge of Information Retrieval in order to define a measure of similarity between the new document and the documents of the training set.

The well - known problem of ”Document versus Query similarity” or ”nearest neighbors” has been the field of continuing research for many years. Various similarity functions / measures, and algorithms have been proposed. A similarity function, adapted from previous works [4, 5], in order to measure the similarity of new (unclassified) documents against the documents of a training set, is the following:

$$S(D_i, D_{new}) = \frac{\sum_{j=1}^{m} q_j k_{ij}}{\sqrt{\sum_{j=1}^{m} q_j^2 \cdot \sum_{j=1}^{m} k_{ij}^2}} = \frac{\sum_{j=1}^{m} q_j k_{ij}}{\sqrt{\sum_{j=1}^{m} q_j^2} \cdot \sqrt{\sum_{j=1}^{m} k_{ij}^2}} = \frac{\sum_{j=1}^{m} q_j k_{ij}}{L_{D_{new}} \cdot L_{D_i}}$$

where m is the number of key-phrases used in the collection, k_{ij} is equal to 1 if the key-phrase j exists in document D_i (of the training set), otherwise is equal to 0 and q_j is the weight of key-phrase j in the new document. The following equation can be used to measure the term q_j:

$$q_j = \log_2\left(\frac{ClassCount}{ClassFreq_j}\right)$$

where $ClassCount$ is the number of classes of the training set, and $ClassFreq_j$ is the number of classes that include the key-phrase j.

The selection of frequent key-phrases is a fundamental aspect of our work. It was influenced by the frequent itemset research and the frequent itemset (episode) algorithms [6–8], but there are the following considerable differences: We look for sequences of words (key-phrases) that will be used as features for classification rules and not for extracting association rules. Consequently, the selected key-phrases should be frequent within the documents (in the frequent itemset context there are transactions not documents) of one or few classes but should not be frequent within the documents of the rest of the classes, while in the case of the fre-

quent itemsets algorithms the selected itemsets should be frequent enough (large, covering, to use the corresponding terminology) in the whole set of transactions.

In the frequent itemset algorithms, transactions are sets of items without any order between them (e.g. products in a shopping basket), while in our case, words that constitute key-phrases are always ordered in the documents.

A third difference is that in the frequent itemset approach the candidate itemsets are not subject to constraints of distance between the items that constitute the itemset, while words (that constitute key-phrases) must always respect distance constraints (coexist in a specific window size, to use the corresponding terminology). Moreover, the window size is not constant but its size is dependent to the number of words that constitute the key-phrase. For example the window size for a 2-word key-phrase could be defined to be 5, while the window size for a 3-word key-phrase could be defined to be 7.

The first of the above mentioned three differences motivated us to work for constructing a new algorithm [9]. The new complete form of our algorithm is presented in section 2. The other two differences drive us to the formation of a new method that creates x-words width key-phrases from frequent (x-1)-words key-phrases. We present this method in section 3 and it is utilized as a step of the algorithm presented in section 2.

2 Selection of Candidate Key-Phrases

The above-mentioned approaches are based on the existence of a list of key-phrases or *Authority List*. The construction of the Authority List will be discussed in this section.

A naive method for Authority List creation is the selection of *sequences of words*, which have high frequencies in the documents of the training set. This is a reliable method for forecasting in general. An adaptation of such a method in a text processing environment can be helpful and especially in the implementation of a *type ahead wizard*. However, *key-phrases (sequences of words)* that exist in many documents of the collection (the whole training set) are not so useful for discriminating between documents.

Another method used for indexing and retrieval of documents [10, 11], chooses indexing terms (stems of words) that exist in a few text but are quite frequent within these texts [12, 13]. The adaptation of this method for selecting key-phrases (sequences of words) could imply some potential problems:

1. A candidate key-phrase that exists in many documents of only one class (and not in another class) could be erroneously rejected if the number of the documents of this class is greater than the number of documents of other classes.

2. A candidate key-phrase could be erroneously chosen if there exists in a small subset of texts of a numerous / dense class and all these texts are dedicated on a specific subtopic of the topic of class.
3. The few documents (texts) that the candidate key-phrase exists could be spread within a lot of classes.

As a first conclusion, the choice of the key-phrases should not be based on frequent (within the whole text collection) candidate phrases neither based on the selection of key-phrases that exist in a few texts of the collection (as a whole) but are quite frequent within these texts.

Instead of using Authority Lists created from the above two mentioned approaches, we can extract key-phrases which are frequent within the documents of one or few classes but are not so frequent in the documents of the rest classes of the training set. We also estimate that the selection of key-phrases based on some syntactic/grammatical structures [14, 15] poses some extra complexity, especially in the case of languages with a rich inflectional system. In such cases, it is necessary to use the morphological parts of speech taggers and / or syntactic analysers.

The above discussion has influenced us in the construction of a new algorithm for key-phrase extraction. We formalise the problem of phrase extraction for classification in the following way:

"Given a dataset (collection) of documents subdivided into classes, a window size (window width) and a frequency threshold, find all key-phrases that occur frequently enough in one or few classes but do not occur frequently enough in other classes". We describe an algorithm for solving this problem. The algorithm has two alternating phases: One for building new candidate key-phrases and another one for evaluating how often these phrases occur in a class of the collection.

The idea of building candidate patterns from smaller ones (see step 7 and step 10 of the algorithm below) is incorporated to the algorithm. Such an idea has been profitably used in the discovery of association rules [6, 7] and occurs also in other contexts [8, 16].

2.1 Algorithm

```
1  For every class (CL_i) of the training set do
2     For every document of the class (DCL_i) do
3        Stemming
4        stopword removal
5     End {For every document of the class}
6     Choose the most frequent stems of the class
      (P_0 parameter)
7     Form the candidate double word phrases (C_2)
```

```
      from the frequent stems (L1)
8     Choose the most frequent double word phrases (L2)
      (W1 and P1 parameters)
9     For x=3 to mpc do
10       Form the candidate x-width word phrases (Cx) from
         the frequent (x-1)-width word phrases (Lx−1)
11       Choose the most frequent x-width word phrases (Lx)
         (Px−1 and Wx−1 parameters)
12    End {For x=3 to mpc do}
13    Compose an integrated list by joining Lx
      (for x=2,3,...,mpc). This join, forms the frequent
      word phrases of class (FCLi)
14 End {For every class of the training set}
15 Integrate / Join the lists of frequent word phrases
   of all classes of the training set
16 Reject the frequent word phrases that exist in many
   classes (Pt parameter). The rest of the frequent word
   phrases form the set of key-phrases or Authority list
17 Form the Dictionary of Terms. It is the list of stems
   that are components of the key-phrases of the Authority list
```

Parameters

mpc	max number of phrase constituents,
P_0	percentage of texts of the class that must contain a stem,
W_1	width of window that covers 2-word phrases,
P_1	percentage of texts of the class that must contain a 2-words phrase,
W_2	width of window that covers 3-word phrases,
P_2	percentage of texts of the class that must contain a 3-words phrase,
...	
W_{mpc-1}	width of window that covers mpc-word phrases,
P_{mpc-1}	percentage of texts of the class that must contain an mpc-words phrase,
P_t	percentage of classes that can contain a key-phrase.

Languages usually have grammatical inflections and consequently words occur in different forms. The need for statistical analysis of texts is related to various processes e.g. to count the frequency of words, retrieve repeated word sequences. Stemming and stop-words' removal are common processes in the field of Information Retrieval [11]. The stemming process or removal of suffixes [17, 18] is used to normalize the word occurrences. The process of stop-words' removal look for words with little contribution to the information communicated with sentences. Hence, articles, prepositions, etc are often regarded as "information noise" and they are removed.

The algorithm works iteratively, alternating between the building phase (step 7 and step 10) and the recognition phases (step 8 and step 11). In the building phase of some iteration ($x = j$), a collection C_j of *new candidate key-phrases* of distinct words is built, using the information available from smaller frequent key-phrases. Then, these candidate key-phrases are recognized in the documents in the class and their frequencies are calculated. The collection L_j consists of frequent key-phrases in C_j. In the next iteration ($x = j+1$), candidate key-phrases in C_{j+1} are built using the information about the frequent key-phrases in L_j. The algorithm starts by constructing C_1 to contain all key-phrases consisting of single words. At the end of each step, the list of frequent key-phrases of the processed class is being built (step 13). At the end, the algorithm composes the Authority list (steps 15 and 16).

The set of candidate 2-words phrases (C_2) must contain key-phrases of length 2 (key-phrases including two stems of distinct words). To construct this set, step 7 forms the Cartesian product and then removes all the tuples that have the same elements. Next section describes the way that step 10 forms the candidate x - width word phrases (C_x) from the frequent $(x-1)$ - width word phrases (L_{x-1}), for every $3 \leq x \leq mpc$.

3 Forming Candidate x-Word Width Key-Phrases From Frequent $(x-1)$-Word Width Ones

The implementation of step 10, of the algorithm in section 2, is presented and briefly discussed in this section.

A naive (brute force) construction of C_x can be based on the calculation of the Cartesian product: $L_1 * ... * L_1$ (x-times). Such a process is obviously time-consuming. Potentially, a large number of candidate key-phrases has also to be checked. Hence, a smart improvement was necessary [9]. The efficiency of a novel algorithm can be implied by the following fact: *"The search space can be drastically reduced if larger key-phrases are formed from smaller ones. In other words, it is only necessary to test the occurrences of key-phrases having sub-portions (key-phrases) that are frequent*".

Such a consideration is influenced by the work on frequent itemset algorithms [6–8], which is characterized by two separate phases. The *generation phase* that combines (joins) couples of frequent itemsets of k size (couples of L_k members) that have $k-1$ common items and produces new candidate itemsets of $k+1$ size (candidate for C_{k+1}); the *prune phase* that removes such candidate $k+1$ size itemsets (candidates for C_{k+1}) that include a k size subset that is not a frequent itemset (not a member of L_k).

As we have already mentioned the frequent itemsets are sets of items without

any order. Thus, if we assume that L_2 contains the itemsets AB, AC, BC and BD, then the generation phase (according to [6]) will suggest as candidates for C_3 the itemsets ABC, ABD and BCD. The prune phase will discard ABD since AD is not member of L_2 and will also discard BCD since CD is not member of L_2. In this example the order of single items (A, B, C and D) inside the itemsets of L_2 is not of any value and the maintenance of itemsets in lexicographic order happens only for technical reasons.

On the contrary, in the case of frequent key-phrases the order of words is a significant feature. Thus, if we use the same example and assume that the letters A, B, C and D represent words, then the generation phase (in our algorithm) should suggest the following, candidates for C_3, key-phrases: ABC, ACB, ABD, BAC, BCD and BDC. In our algorithm there is no need for a prune phase. The rationale is based on the third difference mentioned in section 1. In order to clarify things, we can extend our example and assume that the window size for 2-word sequences (key-phrases) is 5, it is 7 for 3-word sequences and the threshold to accept a word sequence as frequent key-phrase is 3 occurrences. Let assume now that our data (test base) is the following four texts:

```
... A ? ? B ? D ...
... A ? ? ? B ? D ...
... A ? ? ? B D ...
... A ? B ? D ...
```

where

? represents a single word that is not one of A, B or D,

... represents one or more words that neither one of them is A, B or D.

If our algorithm used the prune method then we would have to discard the production ABD, since AD is not included in L_2. This would be a mistake, since the production ABD has 4 occurrences (it is very frequent) in the window of size 7. Thus, in our case, where the window size varies according to the number of words that constitute key-phrases, it is possible a frequent key-phrase to have infrequent sub-key-phrases. Consequently, there is no place for a prune phase in such an algorithm and the method for the creation of x-word width key-phrases from frequent $(x-1)$-word key-phrases must include only a generation phase. We focus on the generation phase in the rest of this section.

Some experimentation was conducted in order to get useful feedback and conclusions related to the extraction of the x-word key-phrases. In the following examples it is supposed that $x = 6$. We present the experimental generation using a graphic way. More precisely, for every example a pair of key-phrases of L_5, is illustrated. It is also described the way of combining the elements of the pair in

order to form new candidate key-phrases (elements of C_6). The pairs of L_5 that we combine have $x-2(=4)$ common constituents (stems). In each key-phrase of the pair there is an unmatched constituent. The unmatched constituents are presented in grey shading. Matching of constituents is represented with arrows.

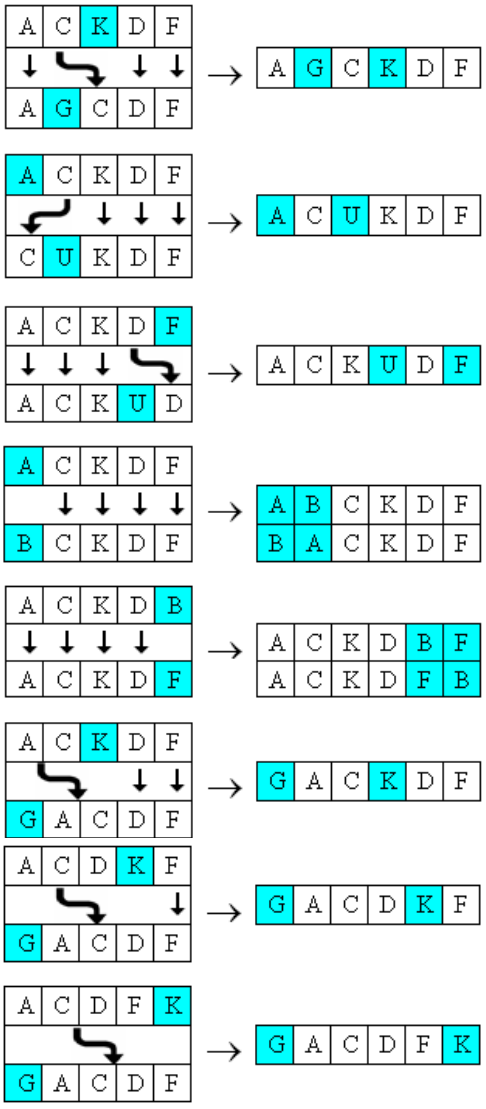

Fig. 2. Generation of C_6.

In the previous examples of generation the elements of C_6 are constructed from elements of L_5. We observe that when two key-phrases have $x-2(=4)$ common

constituents (stems) then every pair of matched constituents have maximum distance one (the constituents are located in the next / previous / same place in the two patterns). On the contrary the unmatched constituents (grey shading) can be found in any distance (from 0 up to $x-2$ places). Also, we observe that as far as we locate the places of unmatched constituents then we can easily produce the new candidate key-phrase (element of C_x).

In order to find if there is a match of $x-2$ constituents (stems) of the elements (key-phrases) we combine, a loop is applied that ”runs” based on two indices (one index for the current stem of each key-phrase). The loop is completed when any one of the indices exceeds its upper limit. For each pair of considered stems, the algorithm checks if they are the same. If it is true the algorithm increases both indices and the counter of matches.

If the considered stems do not match then the indices could have equal values or their difference is one. The whole repetitive form ensures that there is not any case where indices differ more than one.

If the considered stems do not match and the difference of indices is one, then the algorithm stores the smaller index as place of stem (of the corresponding key-phrase) that does not match, and increases the smaller index by one. A question arises, why we do not store the bigger index as place of stem that does match (in the other key-phrase) and increase the bigger indicator by one? This hypothetical action implies that in the next step of repetition the algorithm would consider the matching of two stems with distance two (2). Consequently, this hypothetical action would violate the observation that whenever two key-phrases have $x-2$ common constituents (stems), then every pair of matched constituents has a maximum distance of one. Therefore if the considered elements (key-phrases) have $x-2$ common constituents (stems) then the only way to find this match is to increase the smaller index.

When the considered stems do not match and the indices are equal then the next matching of stems (with respect to the rule of maximum distance one) can be:

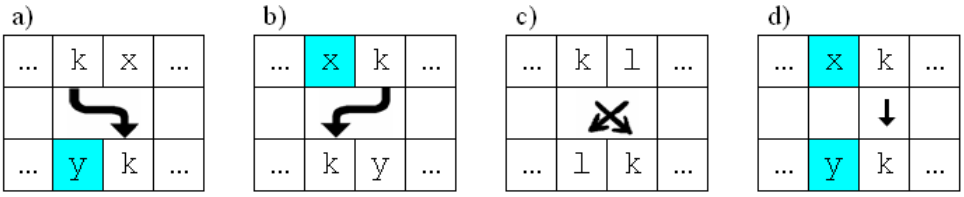

Fig. 3. Possibilities when stems with equal indices do not match.

The algorithm in this case tries (a) and if this case is verified it stores the index of the second key-phrase as the place of stem (of the second key-phrase) that does not match and increases the index of second key-phrase. If (a) is not verified then

the algorithm tries (b). If (b) is verified the algorithm proceeds accordingly. Unfortunately our algorithm does not allow backtracking and it faces the (c) case as if it was (a). Thus, some potential key-phrases of C_x are lost.

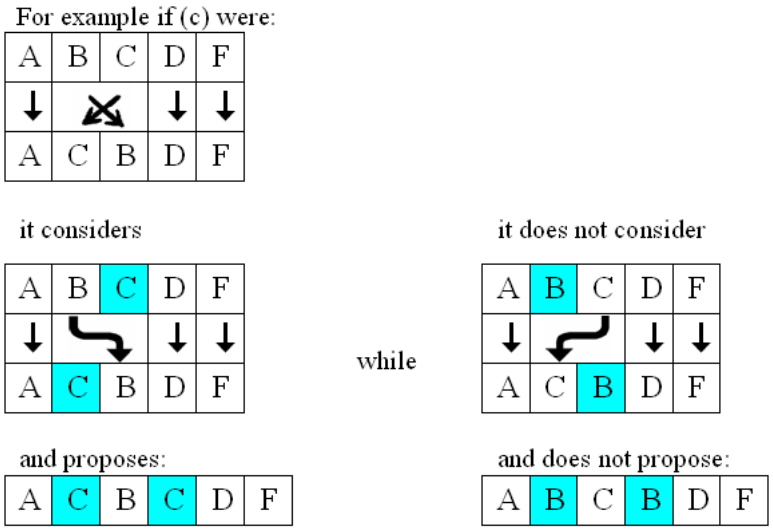

Fig. 4. Potential key-phrases of C_x that are lost.

If the algorithm does not verify, neither (a) nor (b), then the algorithm stores both places as places of stems (of the corresponding key-phrases) that do not match and it advances (by one) both indices. This action postpones, for the next step of repetition, the check for match between stems in the advanced positions.

As we have earlier mentioned this repetitive schema is terminated whenever each one of the indices exceeds its limit. There also exist another case for the termination of the process. For example, when we are considering the matching between the 3rd stem from the first key-phrase ($i1 = 2$) and the 3rd stem from the second key-phrase ($i2 = 2$) then, if the counter of previous matches is at least one (matches ≥ 1) and because it is also possible the considered stems to match, we should continue. As another example, when we are considering the matching between the 3rd stem from the first key-phrase ($i1 = 2$) and the 4th stem from second key-phrase ($i2 = 3$), then, if the counter of previous matches is at least two (matches ≥ 2) and because it is also possible the considered stems to match we should continue. Hence we add the condition: $(matches \geq max(i1, i2) - 1)$.

After the completion of loop, the algorithm considers the counter of matches and if it is $x-2$, then the algorithm examines the places of stems that do not match. If the places are the same the algorithm produces two new candidate key-phrases (elements of C_x), otherwise the algorithm produces one new candidate key-phrase

(element of C_x).

The combinatorial operation of the algorithm, under some circumstances, can lead to the multiple production of the same candidate key-phrase (element of C_x). For example, consider that L_2 is: {AB, AC, BC, CB}, then our algorithm can produce:

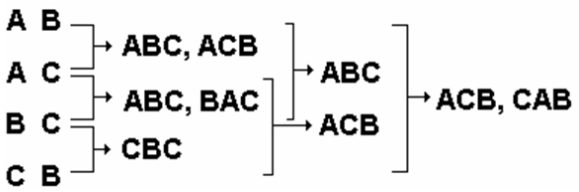

Fig. 5. Production of the same candidate key-phrase.

ABC is constructed three (3) times, ACB three (3), BAC one (1), CAB one (1), CBC one (1) time. For this reason, C_x should be preserved sorted and each new production must be checked before its insertion into C_x.

For the maintenance of list C_x sorted, it is required to traverse the list until it finds a node (key-phrase) greater or equal to the key-phrase under insertion. This traversal requires a lot of string comparisons and consequently has considerable overhead. In order to decrease the comparison's overhead, we could replace stems (strings) with their serial numbers (integers) in the list of highly frequent stems of the class.

4 Conclusions

We have briefly presented two main directions of research for automatic classification of documents based on information retrieval and knowledge discovery in databases. Both approaches are based on the use of key-phrases from a controlled list. Hence, the construction of the key-phrases' list is considered as an important and useful research subject in the domain of document classification. We have presented an algorithm that is based on the idea that the appropriate key-phrases for text classification are those that are frequent within the documents of only one or few classes in the training set. This algorithm reduces the search space by building larger key-phrases from smaller ones.

Acknowledgments

This Project is co-funded by the European Social Fund and National Resources - (EPEAEK-II)-ARXIMHDHS.

References

[1] N. N. Karanikolas and C. Skourlas, "Automatic Diagnosis Classification of patient discharge letters," in *MIE'2002: XVIIth International Congress of the European Federation for Medical Informatics*, Budapest, Hungary, August, 2002, pp. 444–449.

[2] ——, "Naive Rule Induction for Text Classification based on Key-Phrases," in *Proc. of the 6th International Conference on Data Mining, Text Mining and their Business Applications*, Skiathos, Greece, May 2005, pp. 175–182.

[3] N. N. Karanikolas, C. Skourlas, A. Christopoulou, and T. Alevizos, "Medical Text Classification based on Text Retrieval techniques," in *Proc. MEDINF 2003: 1st International Conference on Medical Informatics & Engineering*, Craiova, Romania, Oct. 2003.

[4] N. N. Karanikolas and C. Skourlas, "Computer Assisted Information Resources Navigation," *Medical Informatics & the Internet in Medicine*, vol. 25, pp. 133–146, 2000.

[5] D. Lucarella, "A document retrieval system based on nearest neighbour searching," *Journal of Information Science*, vol. 14, pp. 25–33, 1988.

[6] H. Mannila, H. Toivonen, and A. I. Verkamo, "Efficient algorithms for discovering association rules," in *Proc. KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, Seattle, Washington, July 1994.

[7] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in *Proc. of the 20th International Conf. On Very Large Databases*, Santiago, Chile, Sept. 1994.

[8] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovering frequent episodes in sequences," in *Proc. KDD-95: First International Conference on Knowledge Discovery and Data Mining*, Montreal, Canada, Aug. 1995.

[9] N. N. Karanikolas and C. Skourlas, "Key-Phrase Extraction for Classification," in *MEDICON and HEALTH TELEMATICS 2004: X Mediterranean Conference on Medical and Biological Engineering*, Ischia, Italy, July 31-Aug. 5, 2004.

[10] D. C. Veal, "Techniques of Document Management: A review of Text Retrieval and related technologies," *Journal of Documentation*, vol. 57, pp. 192–217, 2001.

[11] G. Kowalski, *Information Retrieval Systems. Theory and Implementation*. Printed in the USA: Academic Publishers, 1997.

[12] F. Eibe, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning, "Domain-Specific Keyphrase Extraction," in *Proc. International Joint Conference of Artificial Intelligence*, 1999.

[13] I. Witten and F. Eibe, *Data Mining: Practical Machine Learning tools and Techniques with Java implementation*. Morgan Kaufmann, 1999.

[14] B. Georgantopoulos and S. Piperidis, "Automatic Term Extraction Based on Pattern Grammars," LogoNavigation, Issue 57, May 1999.

[15] P. Turney, "Extraction of keyphrases from text: Evaluation of four algorithms," National Research Council of Canada, Technical Report ERB-1051, Oct. 23, 1997.

[16] H. Mannila and H. Toivonen, "Discovering generalized episodes using minimal occurrences," in *Proc. KDD-96: Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, Aug. 1996.

[17] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, pp. 130–137, 1980.

[18] T. Z. Kalamboukis, "Suffix stripping in Greek," *Program*, vol. 29, pp. 313–321, 1995.