Computing and Informatics, Vol. 40, 2021, 1008–1024, doi: 10.31577/cai_2021_5_1008

IMPROVEMENT OF INFORMATION RETRIEVAL SYSTEMS BY USING HIDDEN VERTICAL SEARCH

Suzana Stojković, Nemanja Popović, Ivica Marković

Faculty of Electronic Engineering, University of Niš
Aleksandra Medvedeva 14, 18000 Niš, Serbia
e-mail: {suzana.stojkovic, ivica.markovic}@elfak.ni.ac.rs,
nemanja.popovic@outlook.com

Abstract. The exponential growth of the number of documents in digital libraries and on the Web calls for very intensive development of retrieval systems. One possible architectural approach to IRS, an architecture with hidden verticals, is proposed in this paper. In IRS with hidden verticals, documents from the searched corpus are stored into a predefined set of classes. The user's query is classified before the search, and searching is done only within the corresponding class. The performance of the proposed system is compared to the performance of standard IRS (that contains a unique inverted index) and IRS with cluster pruning (in which searching corpus is clustered and query is compared to the clusters' centroids first, then search is done only in the most similar cluster). Search time in the proposed system is 7.9 times shorter than in the standard IRS and 1.7 times shorter than in the system with cluster pruning. The precision of the proposed system is 2.59 times higher than the precision of the standard IRS, and 1.68 times better compared to the IRS with cluster pruning. The recall of the proposed system is 1.09 times smaller than the recall of the standard IRS, but it is 1.28 times better than the recall of the IRS with cluster pruning. Based on the above results, we can say that proposed approach reduces search time and increases search precision with a minimal reduction in recall.

Keywords: Information retrieval systems, vertical search, classification algorithms, cluster pruning

Mathematics Subject Classification 2010: 94-0

1 INTRODUCTION

Information retrieval systems (IRS) [1] are the systems with the goal to search a large corpus of documents and find the documents that the user needs. The IRS are applied wherever there is a large amount of text documents: in digital libraries, digital encyclopedias (such as Wikipedia), Internet searching services, etc. The corpus that is retrieved by an IRS usually grows rapidly with time. For example, in a Web search, the retrieved corpus consists of all the documents on the Web. New documents are being added to the Web continuously which renders finding the information needed by the user ever harder.

There are three basic requirements that IRS should satisfy:

- The response time should be as short as possible.
- The number of selected documents should not be too large. Experiments from 2006 that are presented in [2] show that 16 % of users of the Web search engines review the first few retrieved documents, 25 % of them review the first page and 27 % review only the first two pages. In [3] statistics from 2020 can be found. It shows that 75 % of the users never scroll past the first page of the search engine results.
- The retrieved documents should satisfy the user's information needs.

To improve all these parameters, vertical search (search on the given domain) [4, 5] and cluster pruning [6, 7] are often used. The idea of both methods is to group the documents from the corpus by any criteria, and then search a single group or several groups only, instead of searching the whole entries corpus.

In a vertical search, the groups are predefined, and the grouping of the documents is done by using classification. Anyone using an IRS with a vertical search should define which group (domain) or set of domains should be retrieved. If the searching is performed only within one domain, it is a domain-specific search. In a cross-domain search, searching is done within the given set of domains.

In IR systems with cluster pruning the groups are not predefined and the end user has no knowledge about the groups. The groups are formed based on similarity among the documents in the corpus by using clustering methods. During the search time, in the first phase, the group (cluster) that is most similar to the query is determined, and in the second phase the determined group is retrieved.

An advantage of a vertical search is that classification methods are usually multiple times faster than clustering. This advantage is especially important when the corpus searched is permanently changing (such is the case in the web searching systems). The advantages of clustering are that a labeled training set of documents is not required and the end user should not need know about the groups. But, if the searching corpus is dynamic, the clustering of the whole corpus should be done periodically because the clusters' centroids should be moved by adding new items.

In this paper, we propose an IRS structure that combines these two methods for improving the performance of the IR system. In the proposed system, a classification method is used for grouping the documents from the corpus. The user query is classified too, and the search is done only in the appropriate group (domain).

The paper is organized as follows. In Section 2, existing structures of the IRS are explained. Section 3 presents the structure of the IRS with hidden verticals. A comparison of the performance of the proposed system with a system without grouping documents in the corpus, and with a system with clustering is shown in Section 4. Section 5 summarizes the results of the proposed IRS structure.

2 IRS STRUCTURES

The information retrieval system accepts user queries in text form, retrieves the corpus of the natural language text documents and returns the list of ranked retrieved documents. In order to speed up the search process, the IRS creates an internal representation of the documents known as inverted index. An inverted index contains data about all the terms in the corpus (in which documents the term appears, how frequently etc.), i.e., the inverted index is a structured representation of an unstructured corpus. Searching for the relevant documents is performed in a structured inverted index, instead of in an unstructured large corpus.

The inverted index can be unique for the whole corpus, or can be divided into partitions corresponding to groups of related documents. Depending on the method of organization of the inverted index, three types of IRS structures can be defined:

- A standard IRS structure that uses a unique inverted index,
- An IRS structure with a vertical search in which the documents are grouped into domains by using a classification method,
- An IRS structure with cluster pruning in which the documents are grouped based on similarity by using a clustering method.

2.1 Standard Structure of IRS

From the observations in the previous section it follows that the major components in an IRS are: the indexer (a component that creates the inverted index of the given corpus) and the search engine (the component that searches the relevant documents in the inverted index). The IRS usually contains a graphical user interface for input of user queries and for displaying the resulting documents. The standard IRS structure is shown in Figure 1.

Additionally, in web searching systems, a web crawler is an obligatory component. A web crawler is a component that traverses the web and collects web pages that make up the corpus to be searched.

2.2 Structure of IRS with Vertical Search

An IRS with a vertical search, or a domain-specific search system, contains a separate index table for each domain. The user defines the query and the domain that will

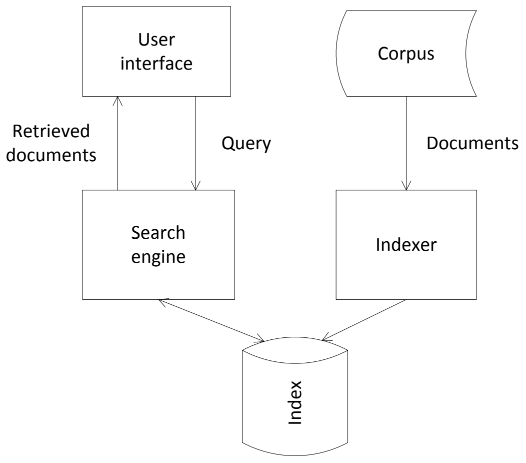


Figure 1. Standard IRS

be retrieved. Domains can be defined by different criteria, depending on the IRS goals. In some search systems, the domains are defined by document types. For example, the most widely used web searcher, Google, supports a vertical search in the following domains: Images, Videos, News, Maps, Books, ... (see Figure 2). Searching by Google can be done without using verticals if the "Web" is selected as the retrieved domain.

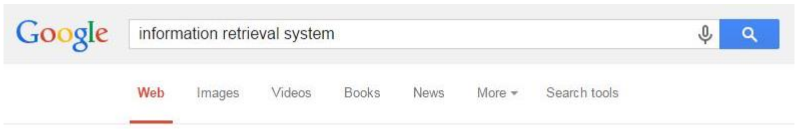


Figure 2. Google's vertical retrieval

Other vertical retrieval systems are systems where the domains are defined by specific topics like: biology, history, computer science, sport etc. In that case, in order to create a different index for each domain, an IRS with a vertical search has to include a tool for document classification into domains. The structure of that type of vertical IRS is shown in Figure 3. For document classification in IRS, any one of the methods for text classification can be used: Naive Bayes, SVM, neural networks, k-nearest neighbors, logistic regression etc. (more on classification methods can be found in [8, 9, 10] and in the references therein).

The vertical IRS always returns only documents from one domain (or from a small number of domains). These systems are realized in one of the following ways:

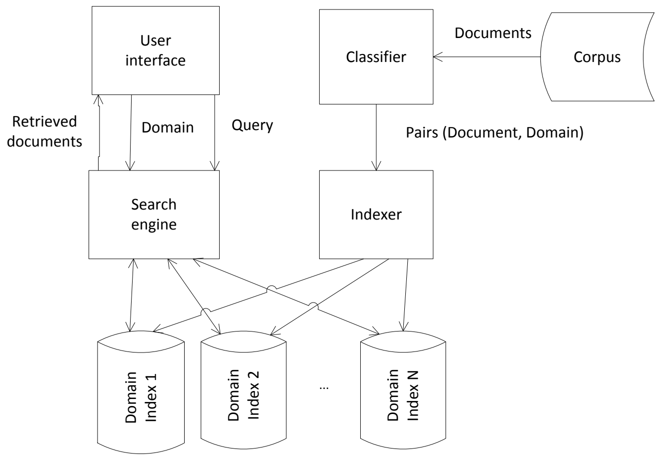


Figure 3. IRS with vertical search

- By classifying documents from searching corpus before or during indexing phase. This approach is frequently used and it is built into certain open source libraries such as Lucene [11] and Solr [12].
- By using an "intelligent" web crawler – a web crawler that collects only documents from the given domain (see for example [13, 14, 15, 16, 17]),
- Retrieval by using a standard IRS and eliminating the irrelevant documents from the returned ones by classification based filters,
- By adding domain-specific keywords to the user's query, and then using the standard IRS for searching (see [18, 19]).

2.3 Structure of IRS with Cluster Pruning

The use of clustering in information retrieval is based on the Clustering Hypothesis: "closely associated documents tend to be relevant to the same requests" [20, 21]. Clustering in an IRS can be used in two ways:

- In the preprocessing phase – assumes clustering the documents in the corpus before searching.
- In the postprocessing phase – assumes clustering the retrieved documents after searching.

In Figure 4 the IRS with cluster pruning (in the preprocessing phase) is shown. In comparison with the system from Figure 2, the Classification unit is replaced by the Clustering unit and the output of this unit are cluster specifications. To create clusters, all the documents from the corpus should be processed. This process is very intensive in terms of computational time and memory. The second disadvantage of this structure is that the cluster specifications should be stored as well. Finally, this method additionally slows down the inverted index update.

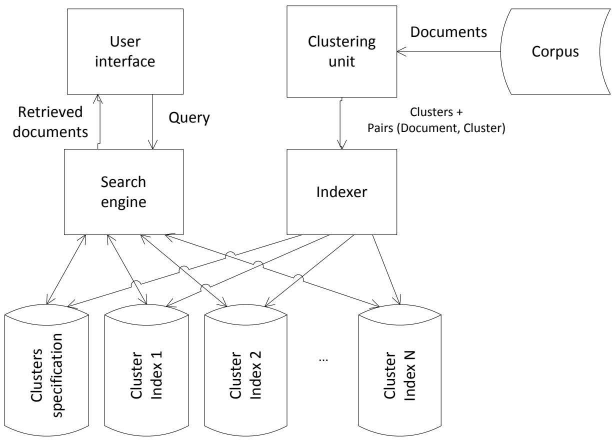


Figure 4. IRS with cluster pruning

Clusters are represented by leaders: centroids [22] or medoids [23] (hypothetical or real documents whose sum of distances to the other documents in the same cluster is minimal). During the search phase, the query is firstly compared to all the leaders, and in the second phase the search is done only in the inverted index of the cluster corresponding to the most similar leader. If the number of clusters is $\sqrt{N}$ (where N is the number of documents in the corpus), similarities between the query and $\sqrt{N}$ leaders are computed, and then the similarities between the query and $\sqrt{N}$ (on average) documents from the appropriate cluster. In this way computational complexity is $2\sqrt{N}$ in total, which is significantly less than N (in a standard IRS, the similarity between the query and all the documents from the corpus should be computed).

Clustering in the postprocessing phase is also discussed in many papers (for example in [24, 25, 26]). The main disadvantage of standard IRS systems is that they occasionally return much more irrelevant documents than relevant ones, a behavior that is oftentimes unavoidable. For example, if the user's query is "jaguar", the IRS will return documents about the animal jaguar, and documents about the car Jaguar. If the results were clustered, one cluster would contain documents about animals, and another about cars, which would enable user to easily decide which documents to read.

3 STRUCTURE OF IRS WITH HIDDEN VERTICALS

The IRS structure with a hidden vertical search is a hybrid structure based on the structure with a vertical search and the structure with cluster pruning. In this structure documents are grouped by a classification method (as in the system with verticals), but the end user is not aware of the verticals (domains). The domain of the query is also determined by the classification method. This can be very helpful because the end user often cannot determine the domain of his query. At times, query classification can be very difficult. For example, many terms are common for documents from the domain of philosophy and documents from the domain of sociology; or for documents about statistics and documents about data mining. In these cases, the search can be done in two or more similar domains. The classifier can return the most likely domain, or the list of possible domains. The structure of the IRS with hidden verticals is shown in Figure 5.

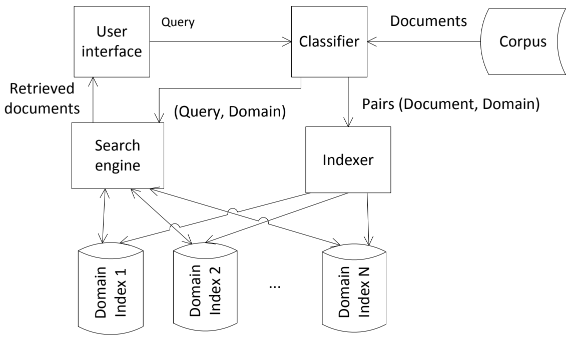


Figure 5. IRS with hidden vertical search

The next question is how to select an optimal classification algorithm among a huge number of existing text classification algorithms (a comprehensive overview of text classification algorithms can be found in [27]), and constantly published new ones (for example [28]). Specificity of the proposed system is that the same method should be used for both document and query classification where queries are extremely short. According to research from [3], the length of the query in 50 % of the cases is up to three words. Similar problem is solved in the research described in [29]. The paper compares the performances of classification algorithms in classification of texts from social networks. Many of these texts are quite short, although not as short as queries. There is no general recommendation which algorithm should be used, but good results are reached by Random Forest and Naive Bayes algorithms. In paper [30] we tested the set of classification algorithms in document and query classification. Results of that research are shown in. In the paper we compared following classification algorithms:

- Random Forest [31],
- Naive Bayes Multinomial [32, 33],
- Bernoulli Naive Bayes [32, 33] and
- SVM [34, 35].

Our experiments showed that a Naive Bayesian Multinomial classifier has about the same results in document classification as an SVM classifier, but it was many times better in a short query classification. Based on the properties of the compared classifiers, we chose the NB classifier for both document and query classification.

4 COMPARATIVE ANALYSIS OF THE IR SYSTEMS WITH DIFFERENT ARCHITECTURES

To test the performance of the proposed architecture, we developed IR systems based on the standard architecture, based on architecture with cluster pruning, and based on architecture with a hidden vertical search. Classification and clustering we used from Weka 3.8.2 library [36]. The classification and clustering model was created by using the StringToWordVector filter that gets a vector with up to 1 000 words from each document. IDFTransform and TFTransform were turned on, and also a stop words list was used. For classification Multinomial Naive Bayesian algorithm is used. For clustering, the K-means algorithm was used. This algorithm uses a distance (dissimilarity) measure between objects for grouping them into clusters. K-means is typically used with a Euclidean distance measure, but for text documents cosine similarity is recommended as a similarity measure. Therefore, we used a custom distance measure calculated by the formula $(1 - \cos\theta)$ which is not implemented in Weka. Indexing and searching was implemented using the Lucene 6.5.0 library [11].

For evaluating the system, we needed a corpus of classified documents and a corpus of classified queries, and for each pair $(\langle document\rangle, \langle query\rangle)$ the relevance of the document to the query needed to be assessed. There are many benchmark corpora for document classification testing, and some corpora of documents and queries for testing information retrieval systems, such as TREC [37], for example. But, there are no known corpora that can be used for both purposes. Because of that, we created the Wikipedia corpus – corpus containing 1225 documents that were taken from the Wikipedia website, and 102 queries suitable for searching in the document corpus. The documents and queries are classified into 10 classes: architecture, art, biology, chemistry, computer science and informatics, literature, mathematics, music, philosophy and physics.

We conducted the experiments on Lenovo W530, the basic parameters of which are shown in Table 1.

First we tested the time and memory complexities of the considered systems. To this end we measured:

CPU	Intel® Core™ i7-3740QM CPU @ 2.70 GHz
RAM	16 GB
OS	Windows 8.1 Professional (x64)
GPU	NVIDIA Quadro K2000M
Hard disc	128 GB SSD-Samsung SM841

Table 1. Experimental system performances

- The creation time of the inverted index including the classification/clustering model creating time (T_{IIM}), and excluding the model creating time (T_{II}). In the case of the standard information retrieval system, those two times are equal because in that case no model is required.
- The memory needed for inverted index storage (M_{II}),
- The average search time for the given set of queries (T_S).

The results of the experiments are shown in Table 2. The rows in the table represent following:

- IRS_S – standard IRS structure,
- IRS_CP10 – IRS structure with cluster pruning by using of 10 clusters. 10 is chosen as the number of clusters because our corpus contains 10 classes.
- IRS_CP6 – IRS structure with cluster pruning by using of 6 clusters. 6 is chosen as the number of cluster because the best retrieving quality (measured by recall, precision and F1 measure) is achieved when the number of clusters was 6.
- IRS_HV – IRS structure with hidden verticals,
- IRS_Type1/IRS_Type2 – ratios of the specified parameters of the IRS of Type1 and the IRS of Type2, i.e.

$$(IRS\_Type1/IRS\_Type2)_x = \frac{IRS\_Type1_x}{IRS\_Type2_x}$$

where x is the observed parameter. In this table:

$$x \in \{T_{IIM}, T_{II}, M_{II}, T_S\}.$$

When all parameters are measured, experiments are repeated 10 times and average values are shown in the table. When searching time is measured, the time of the search of the first query is excluded, because Lucene needed extra time for initialization and searching time of the first query is longer then searching time of the following queries.

From Table 2 we can see that creating the inverted index is the fastest for standard structure. This was expected because in the other two systems the documents should be clustered or classified before indexing.

	T_{IIM} [s]	T_{II} [s]	M_{II} [MB]	T_S [ms]
IRS_S	1.14	1.14	6.27	2.37
IRS_CP10	7.91	2.41	7.41	0.48
IRS_CP6	5.17	2.16	7.08	0.51
IRS_HV	3.95	2.39	7.46	0.3
IRS_CP10/IRS_S	6.93	2.12	1.19	0.2
IRS_CP6/IRS_S	4.54	1.73	1.13	0.22
IRS_HV/IRS_S	3.46	2.1	1.19	0.13
IRS_HV/IRS_CP10	0.5	0.99	1.01	0.62
IRS_HV/IRS_CP6	0.76	1.11	1.05	0.59

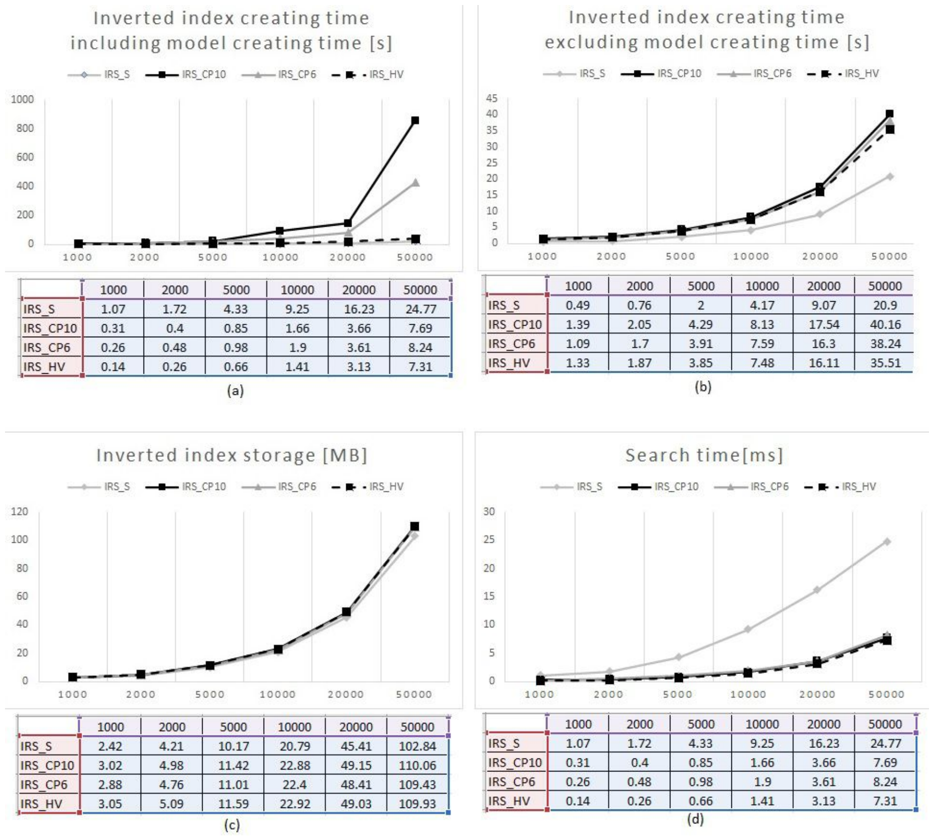
Table 2. Properties of the IRS with different architectures

IRS with cluster pruning and IRS with hidden verticals need clustering/classification model to be created before inverted index generation. As is stated above, in IRS with cluster pruning a new model should be created whenever the corpus changes significantly. Because of that, when IRS with cluster pruning is considered, real inverted index creating time includes clustering model generation. In the case of IRS with hidden verticals, model creating is usually done independently of index generation. The model should be tested before usage, and should not be changed later. In that case, time for inverted index generation should be considered without model generation. However, in order to test the systems under the same conditions, the table contains the times for inverted index generation with and without model generation for both systems. When the model creating time is included, the IRS with cluster pruning is slower. When this time is excluded, inverted index creating times for these two types of the systems are approximately equal. However, if we consider the real use case, inverted index generation time of IRS with hidden verticals excluding model generation time and inverted index generation time of IRS with cluster pruning including model generation time should be compared. In that case, the IRS with hidden verticals is much faster.

When it comes to storing the index, the standard structure requires the smallest amount of storage, while the structure with cluster pruning needs 19 % (13 %) more memory depending on the number of clusters. The structure with hidden verticals uses 19 % more storage than the standard structure. An increase in the used memory space occurs because the inverted index of each cluster (or class) contains its own dictionary, i.e., an almost identical dictionary is stored many times in these cases. Therefore, the inverted index of a system with 6 clusters takes up slightly less memory space. When the number of classes and the number of clusters are equal, index storage in the system with cluster pruning and index storage in the system with hidden verticals are approximately equal.

The parameter more important than the time to create an index and the storage needed for an inverted index is the search time. Search time in the standard structure is just the time of the search in the inverted index. In the structure with cluster pruning this time consists of the time needed to perform the clustering of the search query and to search a specific index. Similarly, in the structure with hidden verticals, this time consists of the time needed to perform a classification with a Naive Bayes Multinomial algorithm and to search in a specific index. In a situation like this, where all the results show the time needed for an end to end search, we can see the structure with hidden verticals performs the best, as it is 7.9 times faster than the standard IRS, and 1.6 or 1.7 times faster than the IRS with cluster pruning (depending whether 10 or 6 clusters are used). The reason for that is the fast classification using the Naive Bayes Multinomial algorithm, and a search done on a small subset of data. From a performance point of view, we can conclude that the structure with hidden verticals is the implementation that performs the best.

Our testing corpus is very small compared to corpora in real IR systems. In order to get an impression of what the relationship of these parameters will be in real systems, we repeated experiments on corpora of 1 000, 2 000, 5 000, 10 000, 20 000 and 50 000 documents. Results of these experiments are shown in Figure 6.



	1000	2000	5000	10000	20000	50000
IRS_S	1.07	1.72	4.33	9.25	16.23	24.77
IRS_CP10	0.31	0.4	0.85	1.66	3.66	7.69
IRS_CP6	0.26	0.48	0.98	1.9	3.61	8.24
IRS_HV	0.14	0.26	0.66	1.41	3.13	7.31

(a)

	1000	2000	5000	10000	20000	50000
IRS_S	0.49	0.76	2	4.17	9.07	20.9
IRS_CP10	1.39	2.05	4.29	8.13	17.54	40.16
IRS_CP6	1.09	1.7	3.91	7.59	16.3	38.24
IRS_HV	1.33	1.87	3.85	7.48	16.11	35.51

(b)

	1000	2000	5000	10000	20000	50000
IRS_S	2.42	4.21	10.17	20.79	45.41	102.84
IRS_CP10	3.02	4.98	11.42	22.88	49.15	110.06
IRS_CP6	2.88	4.76	11.01	22.4	48.41	109.43
IRS_HV	3.05	5.09	11.59	22.92	49.03	109.93

(c)

	1000	2000	5000	10000	20000	50000
IRS_S	1.07	1.72	4.33	9.25	16.23	24.77
IRS_CP10	0.31	0.4	0.85	1.66	3.66	7.69
IRS_CP6	0.26	0.48	0.98	1.9	3.61	8.24
IRS_HV	0.14	0.26	0.66	1.41	3.13	7.31

(d)

Figure 6. Changing of IRS properties by increasing searching corpus

As it shows, the inverted index creating time that includes model creating time exponentially grows with corpus growing in the systems with cluster pruning. If the model is not regenerated, the creating times are approximately equal for all systems with divided inverted index. Memory needed for index storage is changed in the same way in all systems. With the growth of the corpus, difference in storage in the case of the unique and divided inverted index (expressed in percentages) decreases. The greatest improvement in the use of a divided inverted index is reflected in the search time. Difference between search time in standard IRS and in any IRS with divided inverted index increases significantly with increasing the corpus size. Difference between search time in the system with cluster pruning and in the system with hidden verticals becomes negligible with the growth of the corpus because query clustering or classification time becomes much smaller than the inverted index search time.

Finally, the search quality was assessed.

For the Wikipedia corpus of documents and queries, relevant and irrelevant documents were marked and computed recall, precision and balanced F measure (F1 measure) for the search results of each query for different architectures of IRS were computed. The average values of these values are shown in Table 3, and graphical preview of these results is shown in the chart in Figure 7.

	Recall	**Precision**	bfF1
IRS_S	0.879	0.108	0.163
IRS_CP	0.631	0.167	0.232
IRS_HV	0.807	0.280	0.369
IRS_CP/IRS_S	0.72	1.55	1.42
IRS_HV/IRS_S	0.92	2.59	2.26
IRS_HV/IRS_CP	1.28	1.68	1.59

Table 3. Evaluation of IRS systems

As we expected, when the standard IRS is used, the result list contains many documents, but many of them are irrelevant for the query. Because of that, the IRS with the standard structure has the best recall, but the precision is very small. If any architecture with a divided inverted index is used, the number of the retrieved documents is smaller, and the precision is better. By reducing the number of retrieved documents, the number of retrieved relevant documents is reduced, too. It causes a reduction of the system recall.

In the case when the IRS with cluster pruning is used, achieved precision is 1.55 times better, and recall is reduced 1.39 times.

The system with hidden verticals increases precision even more with a much smaller reduction in recall. Namely, the precision of the IRS with hidden verticals is 2.59 times greater than the precision of the standard IRS, while the recall is 1.09 times smaller.

Results of each query were analyzed and for the queries classified correctly the recall of the standard IRS and IRS with hidden verticals turned out to be mostly

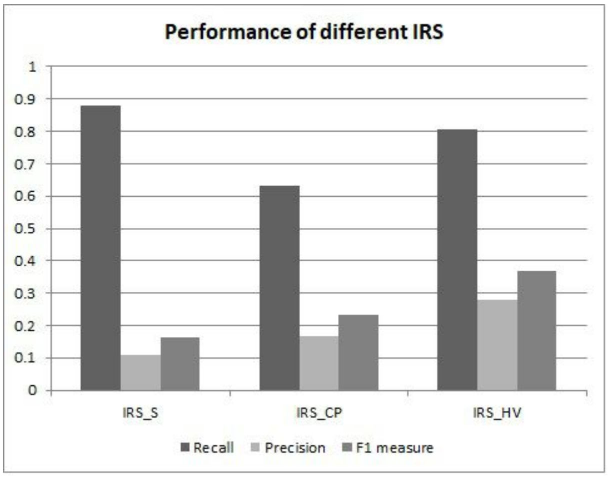


Figure 7. Performance of IRS with different architectures

identical. Table 4 compares the IRS with a standard structure and the IRS with hidden verticals only for queries that are classified correctly. In that case, it can be seen from the table that recalls of both systems are approximately equal, but the precision of the IRS for the system with hidden verticals is 2.61 times better. It follows that the critical point in the proposed system is query classification and it will be an important task for future work. Future research will include different algorithms for documents and query classification. For query classification sereval existing methods for short text classification can be used (such as LSA [38], Bag of concepts [39], etc.), or especially algorithms for query classification (see for example [40, 41, 42]).

	Recall	**Precision**	**F1**
IRS_S	0.883	0.114	0.171
IRS_HV	0.858	0.298	0.393
IRS_HV/IRS_S	0.97	2.61	2.3

Table 4. Evaluation of IRS systems with correctly classified queries

5 CONCLUSIONS

The corpus size in retrieval systems becomes bigger and bigger (specially in web retrieval systems). Because of that, improving the overall performance of an information retrieval system is frequently an issue to be solved. In this paper, we propose an IRS with hidden verticals to improve the search time and precision of the system. Our approach consists of the classification of both the retrieved corpus of documents and the user's query. Documents are classified before indexing, and index tables are created for each class separately. When the user's query is submitted, it is classified too, and the search is done in the corresponding inverted index. In this way, the search space is reduced by a factor approximately equal to the number of defined classes.

We verified the proposed architecture of the IRS by experiments. We compared our approach to other conventional approaches: to the IRS with a unique inverted index (the standard IRS) and to the IRS with cluster pruning (in which documents for searching are clustered and a separate inverted index is created for each cluster).

The results showed that the standard IRS brings the best recall, but its precision is very small and the search time is the worst. The system with cluster pruning brings some improvements in precision and search time, at the cost of reducing the recall. Our approach increases precision, and reduces search time, while the reduction of recall (compared to the standard IRS) is much less. Although our method shows a good performance in the experiments, we believe it can be further improved by increasing the accuracy of the query classification. That should be a topic for our future work.

Acknowledgement

This work is supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

REFERENCES

[1] Manning, C. D.—Raghavan, P.—Schötze, H.: An Introduction to Information Retrieval. Cambridge University Press, Cambridge, England, 2009.

[2] iProspect: Search Engine User Behavior Study. 2006, available at: `http://district4.extension.ifas.ufl.edu/Tech/TechPubs/WhitePaper_2006_SearchEngineUserBehavior.pdf`.

[3] Shelley, R.: 80 SEO Statistics That Prove the Power of Search [2022 Update]. 2021, available at: `https://www.smamarketing.net/blog/80-seo-statistics`.

[4] Chekuri, C.—Goldwasser, M. H.—Raghavan, P.—Upfal, E.: Web Search Using Automatic Classification. Proceedings of Sixth International World Wide Web Conference (WWW6), Santa Clara, California, USA, 1997.

[5] John, B.: The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture. Portfolio, New York, 2005.

[6] Chierichetti, F.—Panconesi, A.—Raghavan, P.—Sozio, M.—Tiberi, A.—Upfal, E.: Finding Near Neighbors Through Cluster Pruning. Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '07), 2007, pp. 103–112, doi: 10.1145/1265530.1265545.

[7] Signitham, P. K. C.—Mahabhashyam, M. S.—Raghavan, P.: Efficiency-Quality Tradeoffs for Vector Score Aggregation. Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB '04), Vol. 30, 2004, pp. 624–635, doi: 10.1016/b978-012088469-8.50056-5.

[8] Duda, R. O.—Hart, P. E.—Stork, D. G.: Pattern Classification. Second Edition. Wiley-Interscience, 2000.

[9] Han, J.—Kamber, M.: Data Mining: Concepts and Techniques. Second Edition. Elsevier, 2006.

[10] Tan, P. N.—Steinbach, M.—Kumar, V.: Introduction to Data Mining. Instructor's Solution Manual. Pearson Addison-Wesley, 2006.

[11] Appache Lucene Core. Available at: `https://lucene.apache.org/core/`.

[12] Apache Solr Reference Guide. Available at: `https://solr.apache.org/guide/`.

[13] Chakrabarti, S.—van den Berg, M.—Dom, B.: Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. Computer Networks, Vol. 31, 1999, No. 11–16, pp. 1623–1640, doi: 10.1016/S1389-1286(99)00052-3.

[14] Sharma, S.: Information Retrieval in Domain Specific Search Engine with Machine Learning Approaches. International Journal of Industrial and Manufacturing Engineering, Vol. 2, 2008, No. 6, pp. 643–646.

[15] Yang, S. Y.: OntoCrawler: A Focused Crawler with Ontology-Supported Website Models for Information Agents. Expert Systems with Applications, Vol. 37, 2010, No. 7, pp. 5381–5389, doi: 10.1016/j.eswa.2010.01.018.

[16] Putra, W. E.—Akbar, S.: Focused Crawling Using Dictionary Algorithm with Breadth First and by Page Length Methods for Javanese and Sundanese Corpus Construction. Procedia Technology, Vol. 11, 2013, pp. 870–876, doi: 10.1016/j.protcy.2013.12.270.

[17] Achsan, H. T. Y.—Wibowo, W. C.: A Fast Distributed Focused-Web Crawling. Procedia Engineering, Vol. 69, 2014, pp. 492–499, doi: 10.1016/j.proeng.2014.03.017.

[18] Oyama, S.—Kokubo, T.—Ishida, T.: Domain-Specific Web Search with Keyword Spices. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, 2004, No. 1, pp. 17–27, doi: 10.1109/TKDE.2004.1264819.

[19] Hoque, M. M.—Poudyal, P.—Goncalves, T.—Quaresma, P.: Information Retrieval Based on Extraction of Domain Specific Significant Keywords and Other Relevant Phrases from a Conceptual Semantic Network Structure. Proceedings of Fifth International Workshop: Forum for Information Retrieval Evaluation (FIRE), India International Center, New Delhi, India, 2013.

[20] Van Rijsbergen, C. J.: Information Retrieval. Second Edition. Butterworths, London, England, 1979.

[21] Hearst, M. A.—Pedersen, J. O.: Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. Proceedings of the 19th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, 1996, pp. 76–84, doi: 10.1145/243199.243216.

[22] Jain, A. K.: Data Clustering: 50 Years Beyond K-Means. Pattern Recognition Letters, Vol. 31, 2010, No. 8, pp. 651–666, doi: 10.1016/j.patrec.2009.09.011.

[23] Kaufman, L.—Rousseeuw, P. J.: Clustering by Means of Medoids. In: Dodge, Y. (Ed.): Statistical Data Analysis Based on the L1-Norm and Related Methods. North Holland, Amsterdam, 1987, pp. 405–416.

[24] Kural, Y.—Robertson, S.—Jones, S.: Clustering Information Retrieval Search Outputs. Proceedings of the 21st Annual BCS-IRSG Colloquium on Information Retrieval (IRSG), Glasgow, Scotland, 1999, pp. 1–9, doi: 10.14236/ewic/IRSG1999.9.

[25] Leuski, A.: Evaluating Document Clustering for Interactive Information Retrieval. Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM '01), 2001, pp. 33–40, doi: 10.1145/502585.502592.

[26] Di Marco, A.—Navigli, R.: Clustering Web Search Results with Maximum Spanning Trees. In: Pirrone, R., Sorbello, F. (Eds.): AI*IA 2011: Artificial Intelligence Around Man and Beyond. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6934, 2011, pp. 201–212, doi: 10.1007/978-3-642-23954-0_20.

[27] Kowsari, K.—Jafari Meimandi, K.—Heidarysafa, M.—Mendu, S.—Barnes, L.—Brown, D.: Text Classification Algorithms: A Survey. Information, Vol. 10, 2019 No. 4, Art. No. 150, doi: 10.3390/info10040150.

[28] Janani, R.—Vijayarani, S.: Automatic Text Classification Using Machine Learning and Optimization Algorithms. Soft Computing, Vol. 25, 2021, No. 2, pp. 1129–1145, doi: 10.1007/s00500-020-05209-8.

[29] Hartmann, J.—Huppertz, J.—Schamp, C.—Heitmann, M.: Comparing Automated Text Classification Methods. International Journal of Research in Marketing, Vol. 36, 2019, No. 1, pp. 20–38, doi: 10.1016/j.ijresmar.2018.09.009.

[30] Popović, N.—Stojković, S.: Analysis of Classification Algorithms for Using in Vertical Retrieval Systems. Proceedings of the 52nd International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST 2017), Niš, Serbia, 2017, pp. 127–130.

[31] Ho, T. K.: Random Decision Forest. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, Vol. 1, 1995, pp. 278–282, doi: 10.1109/ICDAR.1995.598994.

[32] McCallum, A.—Nigam, K.: A Comparison of Event Models for Naive Bayes Text Classification. Proceedings in Workshop on Learning for Text Categorization (AAAI '98), 1998, pp. 41–48.

[33] Metsis, V.—Androutsopoulos, I.—Paliouras, G.: Spam Filtering with Naive Bayes – Which Naive Bayes? Third Conference on Email and Anti-Spam (CEAS 2006), Mountain View, California, USA, 2006.

[34] Cortes, C.—Vapnik, V.: Support-Vector Networks. Machine Learning, Vol. 20, 1995, No. 3, pp. 273–297, doi: 10.1007/BF00994018.

[35] Vapnik, V. N.: The Nature of Statistical Learning Theory. First Edition. Springer, New York, 1995, doi: 10.1007/978-1-4757-2440-0.

[36] Weka 3: Machine Learning Software in Java. Available at: `http://www.cs.waikato.ac.nz/ml/weka/`.

[37] Text Retrieval Conference (TREC). 2016, available at: `http://trec.nist.gov/`.

[38] Dumais, S. T.: Latent Semantic Analysis. Annual Review of Information Science and Technology, Vol. 38, 2005, No. 1, pp. 188–230, doi: 10.1002/aris.1440380105.

[39] Sahlgren, M.—Cöster, R.: Using Bag-of-Concepts to Improve the Performance of Support Vector Machines in Text Categorization. Proceedings of the 20th International Conference on Computational Linguistics (COLING '04), Geneva, Switzerland, 2004, Art. No. 487, doi: 10.3115/1220355.1220425.

[40] Le, D. T.—Bernardi, R.: Query Classification Using Topic Models and Support Vector Machine. Proceedings of the 2012 Student Research Workshop (ACL '12), Jeju, Republic of Korea, 2012, pp. 19–24.

[41] Alemzadeh, M.—Khoury, R.—Karray, F.: Query Classification Using Wikipedia's Category Graph. Journal of Emerging Technologies in Web Intelligence, Vol. 4, 2012, No. 3, pp. 207–220, doi: 10.4304/jetwi.4.3.207-220.

[42] Xia, C.—Wang, X.: Graph-Based Web Query Classification. 2015 12th Web Information System and Application Conference (WISA), 2015, pp. 241–244, doi: 10.1109/WISA.2015.68.

Suzana Stojković is Associate Professor at the University of Niš, Faculty of Electronic Engineering, Department of Computer Science. She is interested in spectral techniques, data mining, natural language processing, and information retrieval.

Nemanjal Popović is Ph.D. student at the University of Niš, Faculty of Electronic Engineering. He is interested in data mining, information retrieval, and business analysis.

Ivica Marković is Teaching Assistant at the University of Niš, Faculty of Electronic Engineering, Department of Computer Science. He is interested in machine learning, data mining, natural language processing, and information retrieval.