ZÁPADOČESKÁ UNIVERZITA

University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
30614 Pilsen
Czech Republic

Document Classification in a Digital Library

PhD Study Report

Jiří Hynek

Technical Report No. DCSE/TR-2002-04
February 2002

Distribution: public

Document Classification in a Digital Library

Ing. Jiří Hynek
jiri.hynek@insite.cz

CONTENTS

1. GENERAL CONCEPTS OF DIGITAL LIBRARIES AND INFORMATION RETRIEVAL

1.1. Introduction

This report is focused mainly on further development of document classification algorithms and their potential applications in various areas of digital library world. Special attention is also paid to clustering and summarization technologies.

Chapter 2 constitutes a description of a real world digital library implemented at a regional power utility company. Chapter 3 is a brief introduction to existing document classification methods, presenting advantages and disadvantages of existing categorization technologies, thus reasoning development of yet another method. *Itemsets categorization*, a new document classification method, is the prime topic of chapter 4 and my further research. Automatic document summarization is the topic of Chapter 5, as this is a prerequisite for *itemsets categorization*. Other potential applications of itemsets method are briefly introduced in Chapter 6.

Development of *itemsets* classifier, a method suitable for short documents, was motivated by presence of freely accessible abstracts on the web. Digital collections of abstracts can be developed cheaply, avoiding the risk of copyright infringement.

My further research work will concentrate namely on the following: domain-independent optimization of *itemsets* categorization method and its application on full-length text files, and testing *itemsets* method on additional document collections (both in Czech and English). Categorization of full-length documents by *itemsets* classifier will require my involvement in research on suitable document summarization methods. I will also look for specific applications of *itemsets* method (and their implementation), namely modification of Naïve Bayes classifier, itemsets-based document clustering, unsolicited e-mail filtering and information querying.

Needed to note that some of the new ideas and concepts contained in this report have already been implemented. Other ideas, suggestions and paradigms will be subject to my further research and, hopefully, also implemented. The prime topic of this report is permanently tested on a real intranet application, which is undergoing quite dynamic day-to-day development.

Numerous information sources on digital libraries and document management systems can be found on the web, such as www.dlib.org (including renowned *D-Lib Magazine*). See Section 8 – Web Information Sources for further references. The issue of digital libraries is also the topic of the well-known *European Conference on Research and Advanced Technology for Digital Libraries* (ECDL).

1.2. Basic Terminology

Let’s define several key terms to be used in the following sections of this report. *Corpus* is an extensive, structured and comprehensive collection of documents in a given language. Corpuses can include either plain or tagged documents. *Morpheme* represents an elementary language element, such as root, prefix, or suffix. *Morphological analyzer* is an automaton providing a set of basic forms (lemmas and grammar rules) for each form of a word. *Morphological normalization* (*stemming*) denotes a process of converting word forms into corresponding basic forms. *Morphological variations* are inflected word forms (word *declensions*) occurring at some languages, such as Czech. *Vocabulary problem* denotes potential use of different synonyms by document authors and users entering their queries. *Phrase* is a short sequence of significant words bearing some meaning. *Phrase search* can be implemented by full-text search of phrases in a document collection. *Stop list* is a list of non-significant words, i.e. words bearing no semantics (prepositions, conjunctions, etc.). *Classification* (or *categorization*) is the process of arranging specific items (such as words, documents, messages) into classes, or categories, using (combination of) features depending on specific classification method. A priori definition of classes (categories) by a librarian is required. *Clustering*, on the other hand, is used to arrange specific items into groups that are defined along the way, without prior definition of categories. An *Itemset* denotes a set of items (such as words, goods in a supermarket, etc.) of some kind.

Various authors often confuse the meaning of *keyword*, *term* and *descriptor*. To be exact, *keywords* represent the simplest expressions (e.g. “liberalization”, “price”, etc.), whereas *terms* and *descriptors* denote multiword language phrases, such as “bipolar transistor” or “structured query language”.

1.3. Quantitative and Qualitative Assessment of the Search Process

In order to assess the quality of a search engine, we need to define some basic parameters and criteria. Elementary measurements include *precision* and *recall* coefficients.

$$\text{Precision}: P = \frac{N_{Rrel}}{N_R} \qquad \text{Recall}: \mathrm{R} = \frac{N_{Rrel}}{N_{DBrel}}$$

Where:

N_{Rrel} = Number of retrieved documents relevant to the user,

N_R = Total number of documents retrieved,

N_{DBrel} = Total number of documents in the document collection relevant to the user.

In order to quantify the above coefficients, users must declare how many documents are relevant to them. It order to determine *recall*, we need to know the number of relevant documents in the whole collection, which is often impossible (such as in case of www).

1.4. Taxonomy of Search Methods

1.4.1. Statistical Analysis of Absolute Term Frequency

Quantification of *absolute term frequency* is the elementary method used by search engines. It entails monitoring the number of matches between the word entered and its frequency in the text database. Such a definition of relevance is quite unreliable, however it is considered sufficient for giving a basic hint. Document *ranking* by search engines is often expressed in percent or by a number. Documents are then sorted according to their ranking, starting with those considered most relevant to the user.

In many languages, such as Czech and other Slavic languages, we are coping with derived word forms (declensions). This is a problem not only for querying, but also for document indexing, namely when we deal with *irregular* word declensions. We must also take into account non-significant words, i.e. words with grammatical function only, lacking any semantics.

1.4.2. Syntactical Methods

Syntactical methods are based on comparing syntactical text structures with structure templates stored in a special database. Basic document structuring is based on document heads, sections, parts, chapters, paragraphs, etc.

1.4.3. Semantic Methods

Document retrieval by means of semantic methods is based on analysis of the semantic structure of document content. Semantic analysis of the text database must be performed. Information retrieval system can be represented, for example, by a semantic tree structure including various topics of interest.

Some search engines display query terms in retrieved documents using different typeface or by means of highlighting (such as *Uniseek* search engine described in section 2.5). Metadata are often displayed in addition to text information in order to provide further semantics.

Full-text search in very large document collections is not very efficient unless it is accompanied by support techniques, such as document structuring, hypertext, thesauri, domain dictionaries, etc. The language of lawyers, for example, is constantly changing by not only introducing new words, but also by altering the semantics of existing terms [22]. Some words, seemingly unambiguous, can have as much as a dozen of different meanings, often contradictory ones. Ambiguity of legal speak is widely known, being less of a problem in strictly technical libraries, such as the one of a power utility company.

1.5. Taxonomy of Search Models

1.5.1. Boolean Model

Boolean model facilitates queries consisting of several words associated by logical operators, as well phrases. By implementing logical operators, we end up with a system based on Boolean logic, i.e. system supporting Boolean search model.

User's query must be passed to lexical and syntax analyzers that must separate terms from logical operators depending on syntax rules, recognize phrases and parenthesized expressions.

Indexing is implemented by means of a sparse Boolean matrix of $k \times n$ (k documents, n terms). Logical operations on terms correspond to Boolean vector operations (using bit arithmetic), with highly efficient processing time. Bitmap indexing can be expressed as follows:

$d_1 (t_{11}, t_{12}, \ldots, t_{1n})$

$d_2 (t_{21}, t_{22}, \ldots, t_{2n})$

…

$d_k (t_{k1}, t_{k2}, \ldots, t_{kn})$

Where $t_{ij} = 1$ iff term j is contained in document i, otherwise $t_j = 0$. For domains of a large cardinality (i.e. high number of terms) we can apply compressed bitmap indexing, using a suitable compression method, e.g. *RLL* encoding.

User's query $\boldsymbol{Q} = (q_1, q_2, \ldots, q_n)$ is represented by a Boolean vector of the length n, which is matched against Boolean vectors of individual documents $d_i = (t_1, t_2, \ldots, t_n)$. For queries containing *AND* operators only (i.e. conjunctive queries, *AND*-queries), d_i is included in the result iff $\boldsymbol{Q}\ \&\&\ d_i == \boldsymbol{Q}$. Should a query include *OR* operators only (i.e. disjunctive query, *OR*-query), document is retrieved iff $\boldsymbol{Q}\ \&\&\ d_i <> 0$. Relevance coefficient can be simply defined by the scalar product of a Boolean query vector $\boldsymbol{Q}$ and the Boolean vector representing document d_i. Documents retrieved upon applying an *OR*-query can be ranked in descending order according to the above scalar product.

In general, Boolean queries can be defined in conjunctive normal form (*CNF*, conjunction of disjunctions of terms or their negations), or disjunctive normal form (*DNF*, disjunction of conjunctions of terms or their negations). The length of a query is then defined as the number of disjunctions (*CNF*), or conjunctions (*DNF*).

A query vector can be subject to query expansion (application of thesaurus and inference rules), converting the original query vector to a modified Boolean vector.

The above Boolean matrix can be refined by specifying exact location of each term in the document. In place of logical values $t_{ij} = 0 / 1$, each matrix cell can represent a record {*0/1; offset*}, where *offset* is the displacement from the origin of the document, according to applicable granularity level (chars, sentences, paragraphs, sections, etc.).

The table below presents customary logical and positional operators and symbols used in definitions of general regular expressions:

Table 1.5.1.-1: **Logical and positional (proxy) operators**

Operator	***Meaning***
X AND Y *X & Y*	Documents containing both term X and term Y (conjunctive query)
X OR Y *X | Y*	Documents containing term X or term Y (disjunctive query)
NOT X *! X* *-X*	Documents not containing term X
X NEAR Y *X ~ Y*	Documents containing term X in the vicinity of term Y (at the distance less than a predefined number of words)
X (n)words Y	Documents containing term X and also term Y at most *n* words after term X
X adj Y	Documents containing term X followed by term Y (the same as *X (0)words Y*)
X sentence Y	Documents containing terms X and Y in the same sentence
X paragraph Y	Documents containing terms X and Y in the same paragraph
(expression)	Expressions in parentheses
"phrase"	Documents containing a specific phrase (several terms delimited by quotes)
+X	Documents that must contain term X

X|category	Documents containing term X at a specific category, e.g. *Delphi|Software*
.	Full-stop represents any character
*x**	Star indicates an arbitrary (also zero) number of occurrences of character *x*
x+	Plus sign indicates arbitrary (1 or more) number of occurrences of character *x*
[s]	Arbitrary character (exactly one) of string *s*
[^s]	Arbitrary character, except chars contained in string *s*
[x-y]	Arbitrary character in the range from *x* to *y*. Several ranges can be defined simultaneously, e.g. *[a-z0-9]* indicates an arbitrary lower case character or number
Other search specifiers or functions	
Language	User can specify the language of document being searched
Family filter	User can leave out documents not suitable for children
Results per page	Users can specify the number of documents per page
Customized settings	E.g. character set, document size, etc.

The disadvantage of the Boolean model is unsatisfactory relevance ranking of documents when displaying query results. Definition of user queries is not intuitive and expressive power of the Boolean model is relatively limited.

1.5.2. Vector Model

Vector model is in fact an extension of the Boolean model, trading logical weights for weight coefficients expressed by real numbers. In vector model we can use more intuitive queries, or even queries in natural language. We are making use of the concept of relevance, which is not covered in the Boolean model. The original system must be enhanced by index tables of significant terms (sparse matrices in a compressed form), weight definitions and efficient index file management. Methods of computing document relevance against query are the issue of proprietary algorithms implemented by search engines.

The following general approach applies:

Let's consider k documents and n index terms. We will assign weights

$w_{ij} = TF_{ij} \times IDF_j$ (the weight of term t_j in document d_i), where:

TF_{ij} = *Term Frequency*, frequency of term t_j in document d_i

DF_j = *Document Frequency*, the number of documents containing term t_j

$$IDF_j = \log\left(\frac{m}{DF_j}\right) + 1$$

IDF_j = *Inverse Document Frequency*, the function inversely proportional to the number of documents containing term t_j, where m is the total number of documents in the collection.

Prior to processing a query, we need to compute query term weights q_j in the range [0;1], by applying the following formula (Salton and Buckley, 1988):

$$q_j = \frac{0{,}5 + (0{,}5 \times TF_j)}{TF_{\max}} \times IDF_j$$

where TF_j is the frequency of query term t_j, TF_{max} is the maximum frequency of an arbitrary query term, and IDF_j represents IDF of term t_j in the document collection.

Index Structure and Similarity

Index is represented by the following data structures: weight matrix W (containing weights w_{ij}), term frequency matrix TF (TF_{ij}), document frequency vectors DF (DF_j), and inverse document frequency vectors IDF (IDF_j). Context (such as headings, text body, keywords) of terms should be taken into account while associating weights with document terms. Indexing granularity can be refined by replacing "pointers" to documents by pointers to specific paragraphs or sentences.

Similarity $Sim\ (\boldsymbol{Q}, d_i)$ can be quantified by various coefficients with subsequent impact on precision (P) and recall (R) of information retrieval.

The simplest (not normalized) method of computing *similarity coefficient* is a plain scalar product of query and document weight vectors:

$$Sim(Q,d_i) = \sum_{j=1,\dots,n} (q_j \times w_{ij})$$

Another popular method is *cosine similarity function*, which can be illustrated by geometrical distance between query and document vectors in vector space of dimension *n*:

$$Sim(Q,d_i) = \frac{\sum_{j=1,\dots,n} (q_j \times w_{ij})}{\sum_{j=1,\dots,n} (q_j)^2 \times \sum_{j=1,\dots,n} (w_{ij})^2}$$

Documents representing result of a query are sorted per relevance, placing first documents with the highest *Sim (Q, d_i)* values.

Conjunctive and disjunctive queries are not distinguished in vector model. *NOT* operator can be implemented by extending the range of query term weights from [0; 1] to [-1; 1].

The main advantage of vector retrieval model as opposed to Boolean model is document ranking per relevance with respect to user's query. Retrieval usability is thus significantly better compared to the Boolean model.

Searching for Similar Documents

The above formula for computing q_j can be used to implement "*find similar documents*" feature. Should it be the case, query is represented by the document (or an abstract) itself – we are trying to quantify similarity between the original document (i.e. the query) and documents in the collection.

Index terms result from the linguistic analysis of all documents in the collection. The number of terms should reflect our effort in reaching compromise between the search speed and full semantic coverage.

1.5.3. Fuzzy Model

Users can specify weights associated with query terms – resulting in a special case of vector search model (*fuzzy* search). Specification of these weights by users is both difficult and subjective.

The advantage of fuzzy concept is the ability to simulate words in natural language, which is extremely important for query languages. Majority of query languages is based on two-value logic *{0;1}*, with no opportunity to convert adequately vague (fuzzy) requirements to the query. The expressive power of the Boolean model is rather limited; therefore we can anticipate increased recall of the search engine by using fuzzy (multiple-valued) logic in the query. Response will include also those items that did not fit into strict two-value criteria. The expressive power of a non-procedural query language containing fuzzy logic elements will therefore increase.

1.6. Linguistic Aspects in the Context of a Digital Library

1.6.1. Morphological Analysis of Queries

A search engine can make automatic query term expansion by synonyms, or possibly other morphological variations of query terms (reverse stemming). Unsophisticated query expansion can, however, result in significant drop in precision (while increasing recall). Further ramification requires user's feedback, often taking place in several query-response phases.

1.6.2. Stemming

Stemming (*lemmatization*, *morphological normalization*) denotes the process of forming basic word forms. Stemming can be used not only for creating normalized index terms, but also for converting query terms into their corresponding base forms. It is common wisdom in IR that stemming improves *recall* with at most a small reduction in *precision*.

The simplest stemming method consists in trivial *cutoff* of word endings (using a database of predefined word endings), so that the resulting word fraction included at least three or four characters. As follows from our practical testing on document collection of Czech technical documents, trivial stemming is quite satisfactory, considering the ease of its implementation. *Dictionary* approach, on the other hand, is based on a brute-force searching in an extensive database of all derived word forms, substituting base forms for corresponding original terms. As stemming is performed off-line, time is not a problem in this case. Quality of stemming clearly depends on the quality of language corpus used. We have implemented dictionary-based stemming using *i-spell* corpus distributed under general public license (GPL). For more information on stemming by means of *i-spell* see section 2.7.

According to Dumais et al. [32] speaking in the context of document classification by inductive learning algorithms, the simplest document representation (using individual words delimited by white spaces with no stemming) was, surprisingly, at least as good as representations involving more complicated syntactic and morphological analysis. However, these results apply to Reuters collection of short English newswire stories. Application to Czech documents leads to largely different results.

Practical stemming algorithms can be found, for example, at http://snowball.sourceforge.net/index.php (stemmers for English, French, Spanish, German, Russian and other languages).

1.6.3. Stop-List

Stop-list (dictionary of non-significant words) is applied to text corpus in order to eliminate words bearing no semantics, i.e. words playing grammatical role only. Application of a stop-list is a must in every digital library. Stop-list used for our digital library (see Section 2) currently contains non-significant English terms listed in the figure below. These terms are used both during the indexing phase and user query processing[1]. A suitable stop-list for the Czech language can be found in [17].

an	the	for	be	is	am	are
you	he	she	it	we	they	them
do	to	in	at	on	if	as
by	of	and	or	then	than	so
which	their	was	were	will	how	when
here	there	not				

Fig. 1.6.3.-1*: Example stop-list for the English language.*

Stop words are removed from text corpus by the lexical analyzer. Collection of non-significant words can be created ad hoc on the basis of a frequency vocabulary, as this task is largely *domain-dependent*. Final content of the stop-list must be fine-tuned manually.

1.6.4. Right-hand Word Expansion

This concept is related to using wildcards in the user query (such as *, %, ?, _). We can modify lexical analyzer to expand query terms with wildcards to corresponding full terms (stored in a database – upon stemming). Search engine is then provided with expanded query without any wildcards. As a result we achieve higher recall (with likely smaller precision).

1.6.5. Thesaurus

Terms contained in a *thesaurus* (a hierarchical dictionary of synonyms) are classified into synonym classes. Thesauri are often used not only for text corpus indexing, but also for information querying. Should a system respond by too few documents, we can apply thesaurus to expand the query by additional terms[2]. By analogy, should the response be too extensive, we can use thesaurus to make query more specific. Systems integrating a thesaurus are sometimes denoted as third-generation full-text search engines. Design of a thesaurus can be simplified by concentrating on a specific domain only.

When expanding user's query, we should take into account the length of the original query: the longer the query, the more synonyms we can add, and vice versa.

Well-defined thesaurus can improve system's response significantly. The system can demonstrate some intelligence, depending on the content and quality of thesaurus database.

[1] Before we eliminate any terms from a query, we should define an irrelevance threshold based on the length of user query.

[2] By using a more general (more specific) term from a thesaurus we can increase recall and reduce precision (increase precision and reduce recall).

Thesaurus can solve, at least partially, the *vocabulary problem*, i.e. use of different terms for the same concept by document authors and digital library users.

Thesaurus can also solve the issue of ever-changing grammar rules, such as concurrent use of Czech terms like “liberalizace” and “liberalisace”, “impulz” and “impuls”, etc.

2. DIGITAL LIBRARY OF ZÁPADOČESKÁ ENERGETIKA

The purpose of this technical digital library is to enhance knowledge and skills of company employees, who should regularly monitor the latest trends and advancements in their area of specialty. Majority of employees does not have the time to monitor information resources. This issue is comprehensively tackled by the intranet digital library, storing information and providing it conveniently to all users. Materials can be stored directly by a person having the information, a designated group of people, or an external data provider.

We will consider a real-life information system called ***InfoServis*** used by Západočeská energetika, a.s., a regional power utility. The system was installed in the commercial environment in early 1999. Author of this report is on the team responsible for the design and development of the library described herein.

2.1. Main Features

Solution is based on a three-tier architecture with thin clients (web browsers) accessing relational database. The system has gradually developed into a full-fledged object-oriented multi-platform client-server application. Text data are stored in a semi-structured object form.

All data in InfoServis are classified into topic trees, which can be defined by a librarian arbitrarily. Tree nodes hold the actual data (called materials). Materials can take various forms, usually consisting of several text fields (e.g. title, short description or an abstract, keywords) and one or more attached files in any format (PDF, MS Word, MS Excel, MS PowerPoint, HTML, XML, etc.).

Main features of InfoServis:

- Document classification using customer-defined taxonomy;
- Immediate access to new materials, quick list of abstracts, full-text available upon request;
- Integration with a search engine (*uniseek*);
- Individual user settings (profiles) – automated notification of new or updated materials;
- Automated replication of data between two InfoServis installations via e-mail;
- All data are treated as objects linked by semantic networks – guaranteeing maximum flexibility and extensibility;
- Automated archiving of outdated documents and corresponding database records.

Integrating InfoServis with *uniseek* (search engine) allows users searching for information in text fields of materials as well as in documents attached. Search can be restricted to sub-trees only. Uniseek is described in detail in section 2.5.

2.2. Technical Features

Implementation is based on a relational database using a tree-like taxonomy. Technical solution includes a non-distributed text database, software tools for graphical database administration using standard web browser, customer-specific taxonomy (based on keyword analysis, customer requirements and domain expert recommendations), and software tools for generating dynamic web pages using C++/Java code.

From users' viewpoint, there are typically no *write* operations (except administrative data); therefore transaction integrity problems are not an issue. Transaction control would be a problem, as web server is a stateless entity (there is no context maintained for individual clients). The SQL server used (*MySQL*) currently does not support transaction processing, which is well balanced by its significant speed in day-to-day usage.

InfoServis can be linked to file server's directory tree to maintain additional information on individual documents. The system can detect new documents and ask their authors to enter the required data.

Software Requirements include *Linux* operating system, *MySQL* database server, *Apache* web server, *PHP 4*, and *Sendmail*. By using MySQL database server we can generate text form of the complete database by single `mysqldump` command, and thus transfer the database to another server easily. By analogy, we can restore the complete database in a batch from a single plain-text file.

Hardware Requirements: Web applications running under Linux are known for modest hardware requirements. InfoServis can run on a modest Pentium II server fitted with 128 MB RAM and IDE hard drive.

Object-oriented design

Each element in InfoServis is treated as an object. Attributes (features) are defined for each object. Object handling depends on the object type (e.g. “article”, “company”, “product”, “process”, etc.). Object types are summarized in the object catalog (see below).

Objects are inter-connected by semantic links (such as *translation* link between two objects of *document* type, *topic* link between object of *document* type and object of *topic* type, *parent* link between two objects of *topic* type, defining tree-like topic hierarchy).

Object catalog

The following major object types are used:

Topics define the area of interest. Topics form a hierarchy, facilitating multi-criterion information classification. *Documents*: Instances of *Document* class represent materials as such (including attributes such as title, language, etc.). *Source:* Source objects are necessary for expanding the information base. We can create links between topics, build taxonomy of sources, create source “rings”, classify sources into topic trees, etc.

Links Catalog

Types of links are defined by means of *links catalog*. It is a list of available links and detailed description of these. Different links are used for different object types; however, some links are used for all objects. We have defined links between topics, between documents, between documents/sources and topics, and between documents and sources.

Object-oriented solution incorporating semantic networks provides opportunity for future development, such as defining inheritance hierarchy of object types (e.g. company – customer – wholesale customer), facilitating various analyses over the data.

Auto-links

InfoServis also includes a special form of references called *autolinks*. These facilitate document classification into tree hierarchy. If we have, for example, an autolink from “Wind power plants” to “Renewable energy”, documents classified to the first class will be also automatically classified into the latter one. It is an example of *unidirectional autolink*. *Bidirectional autlinks* are also utilized widely. Semantically speaking, unidirectional autolinks mostly represent *is-part-of* relationship (such as “e-commerce” topic *is-part-of* “e-anything” topic), whereas bi-directional autolinks represent *is-identical-to* or *is-similar-to* relations, namely when making references from one knowledge tree to another, saving librarian from making multiple classifications.

2.3. System’s Architecture

Implementation of the digital library is based on a three-tier architecture depicted in figure 2.3.-1. Thin client (interface layer) is represented by a standard web browser. Apache web server plays the role of an application server (second layer), communicating via PHP scripts with MySQL database server (third layer). Administrator can also communicate directly with database server by using either command line or *MyAdmin* management tool. Three-tier model is suitable for applications accessing data from various heterogeneous sources, which is the case.

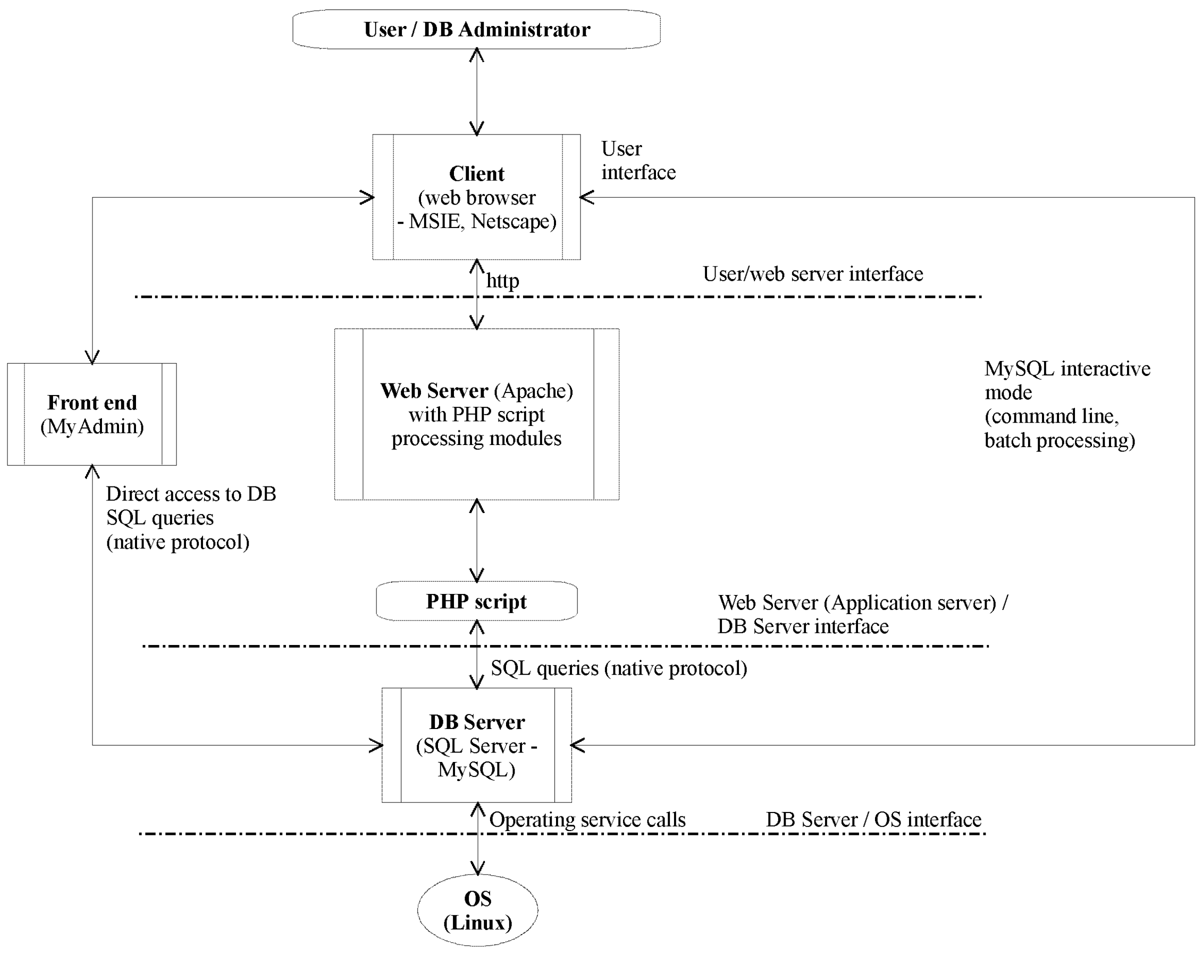


Fig. 2.3.-1: Three-tier architecture of InfoServis.

Flow of documents, parameters and queries is depicted in fig. 2.3.-2. Users working on the Internet cannot add their own documents into the shared library directories.

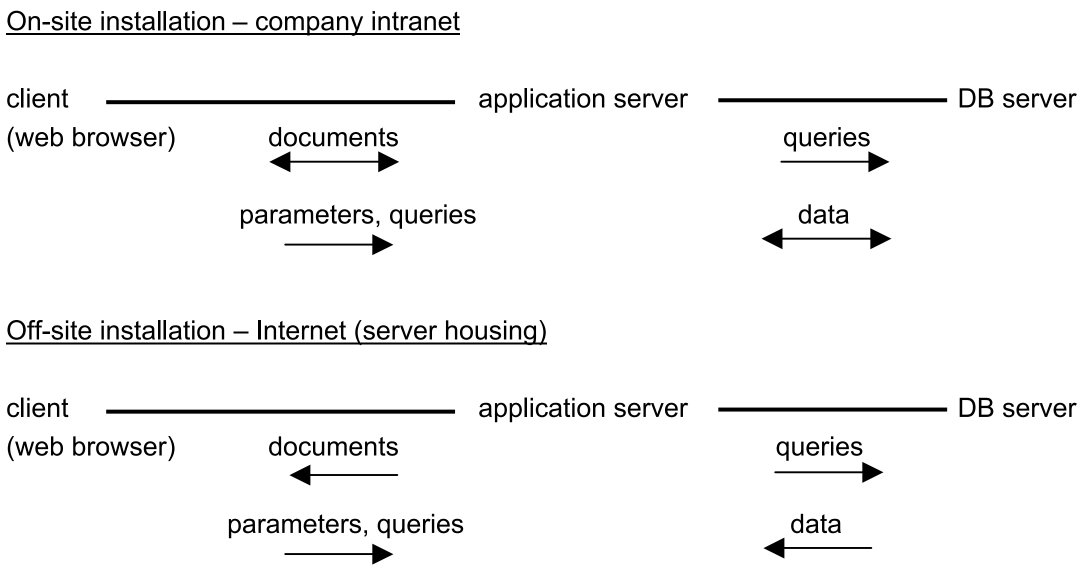


Fig. 2.3.-2: Information flow in the digital library.

Client's role (MSIE, Netscape, Opera) consists in presentation functions only (displaying HTML code distributed by the web server, and sending data entered by users). Platform-independent web application server, Apache, takes care of the business logic (SQL, input data processing and forwarding these data to DB server, management of user profiles, etc.). MySQL database server takes care of the data logic (i.e. applicable data model), data services (input and output of raw data) and file services (operating system calls).

By segmenting the application into presentation logic, business logic and data logic we can achieve much higher flexibility level. Integration of an application server makes connectivity and interoperability among heterogeneous network components much simpler. The middle layer provides for uniform API for all connected clients and the database server.

HTTP transactions between the server and a client take place by opening a connection by the client and sending an applicable URL address. Server gets the request, extracts file requested by the user from the URL address, sends reply information back to the user and closes the connection.

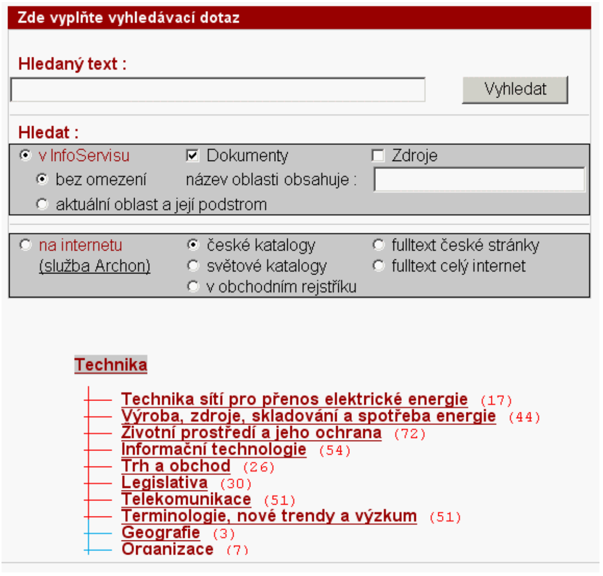
2.4. Querying and User Interface

Users can enter their queries without any a priori knowledge of data structures or location of data being looked up. Users can browse through the topic tree (navigation), optionally displaying all abstracts at the current level and all sub-levels. Navigation results can be ranked per various criteria, such as the date (from the oldest to the most recent), title, users' ranking, or the number of times an abstract was read. Users can also invoke (advanced) full-text search. Search can be invoked globally (in the root of a tree), or at the required topic tree level.

Information search is based on *uniseek* search engine (see section 2.5.) developed especially for multi-lingual digital libraries. It is particularly efficient for east-European languages, such as Czech and Slovak. Readers can use the following search criteria:

- Limiting search to a particular digital library, location or language;
- Limiting search to any topic or combination of topics;
- Exact phrase search;
- Boolean operators (AND, OR, NOT);
- Positional operator NEAR
- Wildcards - *, ?

Entry dialog of the search portal is depicted in the figure below.



Zde vyplňte vyhledávací dotaz

Hledaný text :

Hledat :

[✓ v InfoServisu] [✓ Dokumenty] Zdroje
[✓ bez omezení] aktuální oblast a její podstrom
název oblasti obsahuje :

na internetu (služba Archon)
[✓ české katalogy] světové katalogy v obchodním rejstříku
fulltext české stránky fulltext celý internet

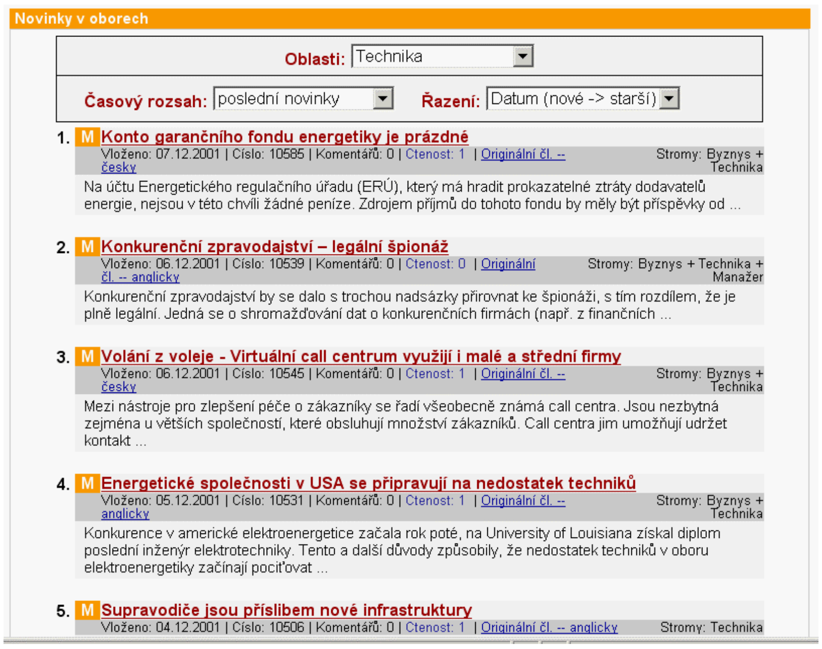
Technika

- Technika sítí pro přenos elektrické energie (17)
- Výroba, zdroje, skladování a spotřeba energie (44)
- Životní prostředí a jeho ochrana (72)
- Informační technologie (54)
- Trh a obchod (26)
- Legislativa (30)
- Telekomunikace (51)
- Terminologie, nové trendy a výzkum (51)
- Geografie (3)
- Organizace (7)

Fig. 2.4.-1: *Dialog box of the search portal.*

Search portal can be further improved by introducing *context-specific search*, i.e. users can specify entities to search in headlines, bodies, or keywords, for example, depending on the structure of documents in the library.

Abstracts are displayed in form of HTML pages generated dynamically from MySQL database. By default, the latest abstracts are displayed first (see the figure 2.4.-2 below).



Novinky v oborech

Oblasti: Technika

Časový rozsah: poslední novinky Řazení: Datum (nové -> starší)

1. M Konto garančního fondu energetiky je prázdné
Vloženo: 07.12.2001 | Císlo: 10585 | Komentářů: 0 | Ctenost: 1 | Originální čl. -- česky Stromy: Byznys + Technika
Na účtu Energetického regulačního úřadu (ERÚ), který má hradit prokazatelné ztráty dodavatelů energie, nejsou v této chvíli žádné peníze. Zdrojem příjmů do tohoto fondu by měly být příspěvky od ...

2. M Konkurenční zpravodajství – legální špionáž
Vloženo: 06.12.2001 | Císlo: 10539 | Komentářů: 0 | Ctenost: 0 | Originální čl. -- anglicky Stromy: Byznys + Technika + Manažer
Konkurenční zpravodajství by se dalo s trochou nadsázky přirovnat ke špionáži, s tím rozdílem, že je plně legální. Jedná se o shromažďování dat o konkurenčních firmách (např. z finančních ...

3. M Volání z voleje - Virtuální call centrum využijí i malé a střední firmy
Vloženo: 06.12.2001 | Císlo: 10545 | Komentářů: 0 | Ctenost: 1 | Originální čl. -- česky Stromy: Byznys + Technika
Mezi nástroje pro zlepšení péče o zákazníky se řadí všeobecně známá call centra. Jsou nezbytná zejména u větších společností, které obsluhují množství zákazníků. Call centra jim umožňují udržet kontakt ...

4. M Energetické společnosti v USA se připravují na nedostatek techniků
Vloženo: 05.12.2001 | Císlo: 10531 | Komentářů: 0 | Ctenost: 1 | Originální čl. -- anglicky Stromy: Byznys + Technika
Konkurence v americké elektroenergetice začala rok poté, na University of Louisiana získal diplom poslední inženýr elektrotechniky. Tento a další důvody způsobily, že nedostatek techniků v oboru elektroenergetiky začínají pociťovat ...

5. M Supravodiče jsou příslibem nové infrastruktury
Vloženo: 04.12.2001 | Císlo: 10506 | Komentářů: 0 | Ctenost: 1 | Originální čl. -- anglicky Stromy: Technika

***Fig. 2.4.-2**: Survey of the latest abstracts in InfoServis.*

The interface has been designed to facilitate full utilization of enterprise document taxonomy, featuring functionality such as intra- and inter-document links, tree browsing, automatic (semantic) inter-topic links, etc.

Data update

Data in the library are updated continuously, as librarians add new items into the collection. Overnight update is performed as well, storing all new documents supplied by information provider. E-mails informing of new items are broadcasted on a daily basis. This model is based on 1:1 information delivery, i.e. one source of information distributes data to one owner of the profile. We can make an enhancement to 1:N model, sending news to all employees of a department, using generic departmental profile.

2.5. Search Engine – Uniseek

The indexing and search server is designed to allow searching defined information sources (within the company or outside). The system features the following characteristics:

- Reading data from various sources (file server, database engines, intranet/Internet, etc.);
- Transferring the data on the indexing server;
- Converting files into a text representation;
- Controlled indexing with saves into a database;
- Searching using a web interface – no need to install any dedicated client software;
- Support for Boolean operators, multiple-word expressions, lexical forms;
- Sorting by relevance, date.

Searching

Search functions are interfaced via a web browser. Users enter their search queries, select individual data locations, and send information to the server. For each object found, a title and a short description are displayed. Selected terms found in the document collection are *highlighted* using various colors, so that user can judge relevance of the reply.

Search time depends on complexity of the search query. Even complicated queries do not take more than several seconds. For a vague query the server can locate thousands of documents – users can gradually refine their queries to find the required information.

System's design

The search engine usually runs on a dedicated machine, reading data across local networks or the Internet. Searching is usually invoked via a web interface. A network client communicating with the search server is available, allowing customers' existing or new applications easily interface into the system – results are generated in form of an XML document.

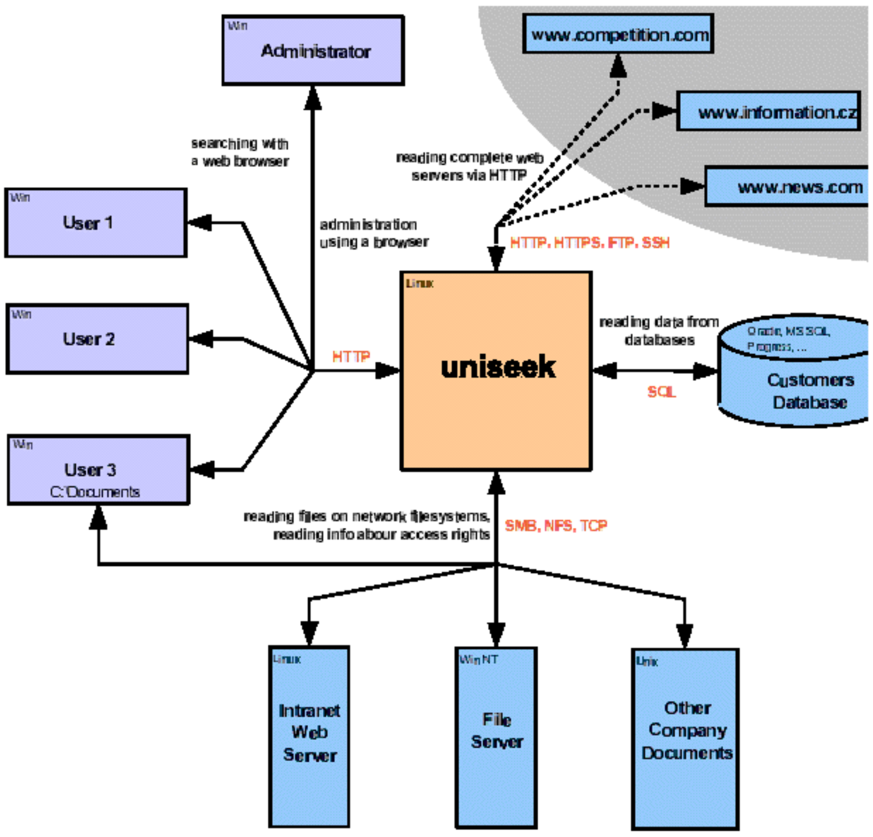


Fig. 2.5.-1: *System's design.*

Indexing

Uniseek can handle multi-language documents. Each language is supported by an extensive dictionary defining lexical relations among individual words (declination, various word forms, singular/plural). The following languages are currently supported:

- Czech (the dictionary contains almost 3,000,000 words);
- English (over 150,000 words);
- German (over 500,000 words).

Since these dictionaries are based on freely available international spell-checker *i-spell*, it is possible to extend support to other languages easily.

During indexing phase, all words are converted into their base form (equivalent of a nominative). Words not covered by the dictionary have their language identified by heuristic algorithms and are converted into their base form using a database of suffices and endings from a given language.

2.6. Linguistic Aspects

The library includes various documents in Czech, English, German and Slovak. Vast majority of documents is either in Czech or English, mostly coming from the web. In order to index and process these documents correctly, we need suitable linguistic tools for both Czech and English. Working with English documents has another advantage: we can compare, for example, classification results with those achieved on Reuters[3]-21578 collection.

It is often the case that a document cannot be classified into a single topic only (in our case this holds true for 8 % of documents only. Most documents are assigned to 3 topics, the average is 2,7, although this number ranges from 1 to 10 (in case of Reuters collection, this parameter ranges from 1 to 16).

[3] Reuters-21578 is currently one of the most popular databases for evaluating text categorization and clustering algorithms.

Stemming and Stop-List Application

Upon application of a stemmer, the number of distinct significant terms dropped by 42 %, with consequent impact on size of database index files. We have used *i-spell* text corpus for Czech stemming, currently containing approx. three million words. All terms are subject to morphological normalization, including terms with irregular declension. As we expected, stemming applied to the English corpus resulted in much less significant drop (20 %).

We have applied both controlled-indexing-based stemmer (*i-spell*) as well as trivial word-endings' cutoff stemmer. Controlled indexing is more complicated, requiring continuous update of the dictionary databases.

By leaving out non-significant words from the index of Czech collection, library volume was reduced by 18 % (in number of terms). Observing this parameter in a long term, there was very little variation, regardless of the total collection volume (always ranging around 20 %). By leaving out five most frequent non-significant terms, the number of terms dropped by more than 10 %. In case of Reuters collection, we have observed drop by 32 %. Impacts of using stop-list have been much more significant for the English language, in spite of using stop-list half the size of the Czech one.

Language	*Stop-list size*	*Number of suffixes used in trivial stemming process*	*I-spell volume (number of terms)*
Czech	484	108	Almost 3 million
English	64	29	500 thousand
German	108	127	150 thousand

According to tests described in [22], a sample of 5,000 full-text documents contained approx. 200,000 various terms (all morphological variations), represented by 80,000 distinct words in their base form (upon stemming).

It is clear that ever-growing volume of the digital library results in adding new technical terms (with constantly slowing speed, as word stock gets saturated), such as chemical substances, foreign words and some special medical terms.

The most frequent significant words in law documents include "law", "republic", "court", "contract", "state", and "legal" [22], as opposed to „system", „electrical", „energy", „market", „device", „control" and „power" occurring in our digital library of a power utility company.

2.7. Further Improvements

Digital library will be enhanced by some innovative and non-traditional functions resulting from research topics constituting focus of this report:

1. *Itemsets* classifier (see Chapter 4) will be optimized to allow automatic categorization of full-length Czech, English, German, and Slovak documents to *InfoServis* topics. The classifier will be also used for *pre-classification* of specific e-mail messages into knowledge-base topics (final classification will be confirmed by a librarian periodically).
2. *Itemsets clustering* (see section 6.1) will be tested on document collection, possibly using it for automatic grouping of discussion group submissions and e-mail messages.
3. Itemsets-based *information push* technology (also see 6.3.2.) will be tested within the context of InfoServis.
4. Search for *similar documents* (also see 6.3.3.) will be further enhanced, possibly using itemsets for this purpose.
5. Results of automatic *document summarization* research will be utilized in order to create document abstracts automatically, as this work currently occupies one full-time employee.
6. I am planning to look for *new features* that can be used for information classification, such as identification of the sender, personal profile of message authors, previous submissions of document originators, keywords, titles, behavioral patterns of *InfoServis* users, pre-defined auto-links, etc.

3. DOCUMENT CLASSIFICATION

3.1. The Task of Document Classification

Classification is used to split data into groups, taking into account specific criteria, attributes, or features. Should these criteria be a priori known for at least a sample of data, we can apply predictive modeling methods and develop a model with classification variable at its output. In the case of text classification, the attributes are words contained in text documents. Feature (attribute) selection is widely used prior to machine learning to reduce the feature space, as the number of features in consideration might become prohibitive.

We are often working with *uncontrolled* classifier, i.e. criteria are not a priori known, making the classifier find these criteria. *Cluster analysis* techniques are applied in these cases.

Learning a classifier (*supervised machine learning*) means inducing a model from the training data set that we believe will be effective at predicting class in new data for which we do not know the class.

In general, we distinguish between *rule-based classifiers* (rules are constructed manually, and the resulting set of rules is difficult to modify) and *inductive-learning classifiers*. Classifiers based on inductive learning are constructed using labeled training data; these are easy to construct and update, not requiring rule-writing skills. In this report I will focus on inductive-learning approach in classifier construction only.

Besides document categorization, we can come across the issue of web page and *link classification*, as introduced by Haas and Grams [37], with useful applications in searching and authoring.

3.2. Existing Document Classification Methods

An interesting survey of five (supervised learning) document classification algorithms is presented by Dumais et al. [32], focusing namely on promising *Support Vector Machines* (SVM) method. *Find Similar*, *Naïve Bayes*, *Bayesian Networks*, and *Decision Trees* methods are also discussed. Another detailed test of text categorization methods is presented by Yang and Liu [33], discussing various algorithms such as *SVM*, *kNN*, *LLSF* (*linear least-squares fit*), *NNet* and *Naïve Bayes*.

Selected existing document classification methods are briefly examined in the table below:

Method	K nearest neighbor (KNN), "Find Similar"
Principle	To classify a new object, find the object in the training set that is most similar. Methods utilizing such principle are sometimes called "memory based learning" methods. *tf*idf* term weights are used, computing similarity between test examples and category centroids. The weight assigned to a term is a combination of its weight in an original query, and judged relevant and irrelevant documents. It is a variant of Rocchio's method for relevance feedback. Cosine value of two vectors (or any other similarity measure) can be used to measure similarity between two documents.
Advantages	Easy to interpret. One of the top performing methods on the benchmark *Reuters* corpus (*Reuters-21450, Apte set*).
Disadvantages	No feature space reduction. Lazy learner – defers data processing until classification time (no off-line preprocessing).
Method	Decision trees
Principle	Model based on decision trees consists of a series of simple decision rules, often presented in form of a graph. These graphs can be quickly modified even by those lacking deep knowledge of statistics. It is a probabilistic classifier – *confidence(class)* represents a probability distribution.
Advantages	Easy to interpret.
Disadvantages	Number of model parameters is hard to find. Error estimates are difficult.

Method	Naïve Bayes (Idiot Bayes)
Principle	Constructed from the training data to estimate the probability of each class given the document feature values (words) of a new instance. Bayes theorem is used to estimate these probabilities. It is a probabilistic classifier – *confidence(class)* represents a probability distribution.
Advantages	Works well even when the feature independence assumed by Naïve Bayes does not hold. Surprisingly effective.
Disadvantages	Simplifying assumptions (conditional independence of words).
Method	Unrestricted Bayesian classifier
Principle	Assumption of word independence is not applied. Its alternative – semi-naïve Bayesian classifier – iteratively joins pairs of attributes to relax the strongest independence assumptions.
Advantages	Simple implementation, easy interpretation.
Disadvantages	Exponential complexity due to assuming conditional dependence of words.
Method	Neural networks (perceptrons)
Principle	Separate neural network per category is constructed, learning a non-linear mapping from input words (or more complex features, such as itemsets) to a category.
Advantages	Design is easy to modify. Various models can be constructed quickly and flexibly. Subject to intensive study in artificial intelligence.
Disadvantages	Model based on neural networks does not provide any clear interpretation. High training cost (more time consuming than the other classifiers).
Method	Linear SVM
Principle	An SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin. The margin is defined by the distance of the hyperplane to the nearest of the positive and negative examples. SVM (optimization) problem is to find the decision surface that maximizes the margin between the data points in a training set.
Advantages	Good generalization performance on a wide variety of classification problems. Good classification accuracy, fast to learn, fast for classifying new instances.
Disadvantages	Not all problems are linearly separable.
Method	Itemsets Modification of Naïve Bayes – see Section 6.2.

Development of yet another classification method was motivated by the need of *processing short-documents* (abstracts). It is likely that size of digital libraries will increase rapidly in the near future and proper classification of abstracts will become even more important. Efficiency of universal document categorization methods will gradually decrease, necessitating ad-hoc classification methods, such as *itemsets*, offering easy adjustment to a particular document collection.

3.3. Test Document Collections

Classification algorithms are tested on large sample document collections in order to assess their viability and compare one to another. *Reuters-21578* collection[4] is gaining popularity among researchers in text classification[5], becoming a widely used benchmark corpus. Collection includes short newswire stories in English, classified into 118 categories (e.g. *earnings*, *interest*, *money*, etc.). Each story is assigned to 1.3 categories on the average (maximum is 16); however, there are many unassigned stories as well. Original classification is highly skewed, as a very large portion of stories is

[4] Publicly available at http://www.research.att.com/~lewis/reuters21578.html

[5] Topic spotting for newswire stories is one of the most commonly investigated application domains in text categorization literature.

assigned to *earnings* category only. Top 10 categories include 75 % of all instances, and 82 % of the categories have less than 100 instances.

We have also made various tests on our proprietary collection of technical documents from the digital library of a power utility (more than 4,000 text documents in Czech, English, German and Slovak). There are other widely used collections, such as MEDLINE[6] (medical texts in English), UCI ML Repository[7], or Czech national corpus[8].

The size of test collection is an important issue. When testing a classification algorithm, we need to examine how many positive training examples are necessary to provide good generalization performance. According to Dumais et al. [32], twenty or more training instances provide stable generalization performance[9]. According to Yang and Liu [33], *SVM* and *kNN* classifiers significantly outperform *NNet* (*neural nets*) and *Naïve Bayes* when the number of positive training examples per category are small (less than ten). The required number of training examples is therefore specific for each classification algorithm.

As opposed to testing classification algorithms on collections of short abstracts, various tests on full-text document collections have been performed, such as the one by Beney and Koster [38], testing *Winnow* classifier[10] on patent applications supplied by the European Patent Office (documents about 5,000 words each).

3.4. Assessment of Classification Algorithms

Classification algorithms are evaluated in terms of speed and accuracy. Speed of a classifier must be assessed separately for two different tasks: *learning* (training a classifier) and *classification* of new instances.

Many evaluation criteria for classification are proposed. *Precision* and *recall* criteria are mentioned most often. *Break-even point* is proposed by Dumais et al. [32] as an average of precision and recall. Decision thresholds in classification algorithms can be modified in order to produce higher precision (at the cost of lower recall), or vice versa – as appropriate for different applications. Averaged F_1 measure[11] is commonly used for classifier evaluation. Single valued performance measures (p, r, F_1) can be dominated by the classifier's performance on common categories or rare categories, depending on how the average performance is computed [33] (*micro-averaging* vs. *macro-averaging*).

In the case of mono-classification, some researchers (e.g. [38]) report *error rate* measure, which is percentage of documents misclassified.

Yang and Liu [33] report an interesting controlled study with statistical significance tests on five text categorization methods. As categories typically have an extremely non-uniform distribution (such as the case of *Reuters-21578*), it is meaningful to assess a classifier with respect to category frequencies. With respect to *Reuters-21578* benchmark corpus, *ApteMod* version is often used, which is obtained by eliminating unlabeled stories (i.e. unclassified instances) and selecting the categories which have at least one document in the training set and the test set.

It is important to note that classifier's performance largely depends on splitting the corpus on training and testing data. Testing the classifier on training data used for learning the classifier often leads to significantly better results.

The problem with evaluating classifiers is their domain dependence. Each classifier has a particular sub-domain for which it is most reliable [35]. In order to overcome this issue, multiple learned classifiers are combined to obtain more accurate classification. Separating the training data into subsets where classifiers either succeed or fail to make predictions was used in Schapire's Boosting algorithm [36]. A decision tree induction algorithm (such as *C4.5*) can be trained and applied to

[6] MEDLINE is available at: http://www.nlm.nih.gov/databases/databases_medline.html

[7] UCI Repository of Machine Learning databases, 1996, available at: http://www.cs.uci.edu/~mlearn/MLRepository.html

[8] Available at: http://uckn.ff.cuni.cz/CZ/cnc

[9] We have resorted to at least 50 positive training examples per category while testing *itemsets* classifier.

[10] Description of the *Winnow* algorithm can be found at http://www.cs.kun.nl/doro

[11] F_1 was initially introduced by van Rijsbergen [34]; it is defined as: $F_1(r,p) = 2rp / (r+p)$

distinguish between cases where the classifier is correct and those where it is incorrect.

4. ITEMSETS CLASSIFIER

4.1. Association Rules

The topic of association rules is of key importance for many document classification algorithms. Association rules, and the related concept of itemsets, constituted my motivation for developing a new document classifier.

It is imperative to find a method for automatic generation of association rules over the word domain. We can start with keywords, checking which word pairs are present in more than τ documents (τ is a threshold value), or possibly documents classified to a specific topic. We can also look for associations among terms with specific semantics (semantic tags can be supplemented by means of a special lexical analyzer). We can define weights of association rules by frequency of occurrence, or possibly by distances between terms in an association. Association among terms cannot be regarded as causality relationship, as we do not know the direction.

4.2. Itemsets Classifier

Original classification method, called *itemsets classification*, has been developed to facilitate automatic classification of short-documents in the digital library of Západočeská energetika. Majority of traditional document classification methods is based on repeated word occurrence, which is impractical to use in case of very short documents (less than 100 significant words in this case).

Our aim was to produce a taxonomy reflecting information-seeking behavior of a specific user population, i.e. employees of a regional power utility. Functionality of the digital library simplifies creation of enterprise information portal (EIP). Automatic classification (auto-categorization) engine facilitates continual monitoring of the information generated by the company (or external sources) and organizing it into directories.

Itemsets method resulted from our basic assumption, that objects belonging to the same concept (class) demonstrate similar features. Learning paradigm based on such an assumption is called *similarity-based learning*. Objects representing instances of the same concept constitute "clusters" in the concept space. The task of modeling is to assume finite number of instances and find general representation[12] of these clusters, which we call *classes*, *topics*, or *categories*. Classification algorithm looks for knowledge that can be used for classifying new instances.

We are using inductive inference based on *inductive learning hypothesis*: Any hypothesis found to approximate the target (classification) function well over a sufficiently large set of training examples (abstracts in the training set, in this case) will also approximate the target function well over other unobserved examples (abstracts to be classified).

Itemsets method is robust to errors (alike decision tree learning methods) – both errors in the classification of training documents (made by a librarian manually) and errors in the attribute values (significant terms) that describe these documents.

Abstracts of technical articles are mostly freely accessible on the web. It is therefore possible to create an extensive library of these abstracts. Users of the library can then make a request to buy a full copy of a document or its translation. The task of document searching in the digital library is similar to the one of categorization, being solved by means of similar principles.

4.3. Apriori Algorithm for Itemsets Generation

The *apriori algorithm* (Agrawal et al.) is an efficient algorithm for knowledge mining in form of association rules [25]. We have recognized its convenience for document categorization. The original apriori algorithm is applied to a transactional database of market baskets. In the context of a digital library, significant terms occurring in text documents take place of items contained in market baskets (itemsets searching is equivalent to term clustering process) and the transactional database is in fact a set of documents (represented by sets of significant terms). Consistently with the usual terminology let's denote terms as items and sets of items as *itemsets*.

[12] Herein below denoted as „characteristic itemsets".

Let π_i is an item, $\Pi = \{\pi_1, \pi_2, \dots, \pi_m\}$ is an *itemset* and Δ is our *document collection* (representing *transactions*). The itemset containing k items is called *k-itemset*. Each itemset Π is associated with a set of transactions $T_\Pi = \{T \in \Delta \mid \Pi \subseteq T\}$ which is a set of transactions containing itemset Π. Frequency of an itemset is defined as a simultaneous occurrence of items in data in consideration. Within our investigation we often utilize the threshold value employed for the minimum frequency (*minsupport*) of an itemset. *Frequent itemsets* are defined as those whose support is greater than or equal to *minsupport*. The (transaction) *support* in our case corresponds to the frequency of an itemset occurrence in the (transaction) database Δ ($\Delta = \{T_1, T_2, \dots, T_n\}$, T representing transactions). The support *supp(Π)* of an itemset Π equals $|T_\Pi| / |\Delta|$. Support is defined over the binary domain $\{0, 1\}$, with a value of 1 indicating the presence of an item in a document, and the value of 0 indicating absence of an item (frequency of an item is deemed irrelevant, as opposed to traditional TF×IDF methods)[13]. Support fails as a measure of relative importance whenever the number of items plays a significant role. We have found that this is not a problem for short-document classification task.

Declaring itemsets *frequent* should they occur in more than *minsupport* number of documents in a particular class is correct[14], with no need of normalization like in case of IDF concept (we do not eliminate those in the upper frequency range, as itemsets are to characterize a class based on their frequent appearance). At this phase we have already eliminated stop (non-content) words; moreover, we are deciding on „being frequent" within the scope of a class, not the whole document collection. Co-occurrence of terms representing an itemset is domain-restricted, domain being a class (category).

Our goal is to discover frequent itemsets in order to characterize individual topics in the digital library.

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

Let's suppose we have TD_S distinct significant terms in our document collection Δ. Firstly we generate candidates of frequent 1-itemsets (shortly „candidate 1-itemsets"). These are stored directly in index files in *DF* (`Document Frequency`) table. Consequently, we compute frequent 1-itemsets. In the next step, we generate 2-itemsets from frequent 1-itemsets. Generation of subsequent candidate and frequent *n*-itemsets continues until the process of frequent itemsets' searching terminates with regard to apriori property ("all non-empty subsets of a frequent itemset must be frequent"). While implementing this method, we utilize a technique similar to *transaction reduction* method: a document that does not contain a k-itemset can be left out of our further consideration, since it cannot contain any of (k+1)-itemsets.

Let $Ç_k$ denote a set of candidate k-itemsets and F_k a set of frequent k-itemsets. Generation of F_k from F_{k-1} is based on the following algorithm (our modification of the original Apriori algorithm by Srikant and Agrawal):

```
// For 1-itemsets:
$Ç_1$  :=  all significant terms in $\Delta$;
$F_1$ := $\emptyset$;
for $\forall$ $\Pi_i \in Ç_1$ do
  for $\forall$ $t_j \in T$ do
    if (supp($\Pi_i$) in class $t_j$ is greater than or equal to minsupport)
      then begin
        add $\Pi_i$ to $F_1$
        break;
      end;

// For k-itemsets, where k > 1:
$F_k$ := $\emptyset$;
for $\forall$ $\Pi_i \in F_{k-1}$ do
  for $\forall$ $\Pi_j \in F_{k-1}$ do
```

[13] In case of a shopping-basket transaction database, support provides a misleading picture of frequency in terms of the quantity of items sold. This is not the case of document collection, taking documents as transactions.

[14] The *class support* is a variation of the usual support notion and estimated the probability that the itemset occurs under a certain class label.

```
    if (the first k-2 items in $\Pi_i$ and $\Pi_j$ equal, but the last items differ)
      then begin
        c := $\Pi_i$ join $\Pi_j$;
        if ($\exists$ subset s, s $\subset$ c having k-1 elements, where s $\notin F_{k-1}$)
          then break;
          else for $\forall$ t $\in$ T do
            if (supp(c) in class t is greater than or equal to minsupport)
              then begin
                add c to $F_k$
                break;
              end;
             end;
```

4.4. Document Classification Using Itemsets

The following notation will be used in this section:

Π	Frequent itemset	$D\Pi_i$	Set of documents containing itemset Π_i
$\lvert\Pi\rvert$	Cardinality of frequent itemset Π	$\lvert D\Pi_i\rvert$	The number of documents containing the itemset Π_i
T	Topic (representing a categorization class)	DT_i	Set of documents associated with topic T_i
D	Document	$\lvert DT_i\rvert$	The number of documents associated with topic T_i
$\overline{D}$	A set of significant terms contained in document D	$\lvert T\Pi_i\rvert$	Number of topics in which itemset Π_i is frequent
L N_i	The number of topics The number of frequent itemsets of cardinality i	C_i $\lvert C_i\rvert$	Set of itemsets characterizing topic T_i The number of itemsets characterizing topic T_i

The Classification Problem

The classification problem can be divided into two parts: *training phase* and *classification*[15] *phase*. The training phase consists of the following:

- Define a set of topics (categories) by a domain expert[16]. *L* categories are thus defined.
- Insert (manually) a certain number of documents into topics, i.e. classification attributes are defined for each class (training data set). A domain expert performs categorization of all available training documents. Each topic should be assigned a statistically significant number of documents.
- Automatic generation of representative itemsets of different cardinality for each topic.

While performing classification, we utilize representative (characteristic) itemsets to classify documents into corresponding topics.

The classification algorithm can be evaluated in terms of accuracy (*precision* and *recall* parameters) and speed. Accuracy can be measured by means of a test-set, the members of which have a priori known classification. *Precision*: $P = p/q$; *Recall*: $Q = p/r$, where p is the number of classes determined correctly by the classifier (automatically); q is total number of classes determined automatically; r is the number of classes determined by a domain expert (manually, i.e. correctly).

Note the analogy with corresponding parameters defined for search engines as well as association rules in databases of business transactions:

[15] A classifier is a function mapping a vector of document terms onto a set of topics (classes): $f(\overline{D})=\{topics\}$

[16] Categories are often represented in a hierarchical form. We are not considering this aspect in our research, although it would not be a problem – we might combine leaf topics into a single node, retraining the classifier on the adjusted structure.

Classifier	Search engine	Association rules
Precision $P = p / q$	Precision $\frac{\textit{found relevant}}{\textit{found total}}$	Confidence $Conf\,(X \Rightarrow Y) = P(Y \mid X) = \frac{\textit{number of transactions } X \cup Y}{\textit{number of transactions } X}$
Recall $Q = p / r$	Recall $\frac{\textit{found relevant}}{\textit{relevant in database}}$	Support $Sup\,(X \Rightarrow Y) = P\,(X \cup Y) = \frac{\textit{number of transactions } X \cup Y}{\textit{number of all transactions}}$

Where *number of transactions* $X \cup Y$ denotes the number of transactions containing both X and Y.

We are computing overall precision and recall parameters using *micro-averaging* method, i.e. we are giving equal weight to each object (rather than each category, as in *macro-averaging*). P and R are computed for each new test document separately, then figuring out the average of all precision and recall values over the whole test set.

Quality of classification is often expressed by means of *F-measure* (or *F-score*):

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

α representing relative importance attributed to precision P.

The following F-measure is often used in the context of text retrieval and classification (by substituting α = ½), denoting it as F_1-measure (see also 3.4):

$$F = \frac{2 \times P \times R}{P + R}$$

Complexity of itemsets classification algorithm

Complexity of classifying one document can be expressed approximately as $C_{AVG} \times L \times K$ (= $C_{TOT} \times K$), where C_{AVG} is the average number of itemsets in C, L is the number of classes, K is a constant representing average complexity of comparing itemsets in C with document being classified[17] and C_{TOT} is the total number of itemsets in all C files. Time requirements of classification as such are relatively low compared to complexity of the training phase.

4.5. Phases of Itemsets Method

4.5.1. Training Phase

The *training phase* can be also described as *feature selection phase* (reduction of the feature space). For each itemset Π_j we can find a corresponding set of documents containing Π_j. Let's designate this set of documents as $D\Pi_j$. It is obvious that cardinality of $D\Pi_j$ will be higher than a certain threshold value, since Π_j was selected as a frequent itemset.

By analogy, for each topic T_i there is a characteristic set of documents associated with this topic. Let's designate this set as DT_i. Altogether we will have L sets.

Our goal is to specify a certain number of itemsets for each topic, where each itemset is associated with a subset of the set of topics. Namely, itemset Π_j is associated with topic T_i corresponding to the values of $w_{\Pi_j}^{T_i}$ exceeding some threshold value. The weight of $w_{\Pi_j}^{T_i}$ can be computed as follows[18]:

$$w_{\Pi_j}^{T_i} = \frac{\left|D\Pi_j \cap DT_i\right|}{\left|DT_i\right| \times \left[1 + \left|D\Pi_j\right| - \left|D\Pi_j \cap DT_i\right|\right]} \qquad i = 1, 2, \ldots, L$$

[17] K naturally depends on the size (in the number of terms) of document being classified. Needed to note that twice long a document does not mean twice longer the classification. We are working with significant terms only (leaving out stop words), also neglecting repeated occurrence of significant terms.

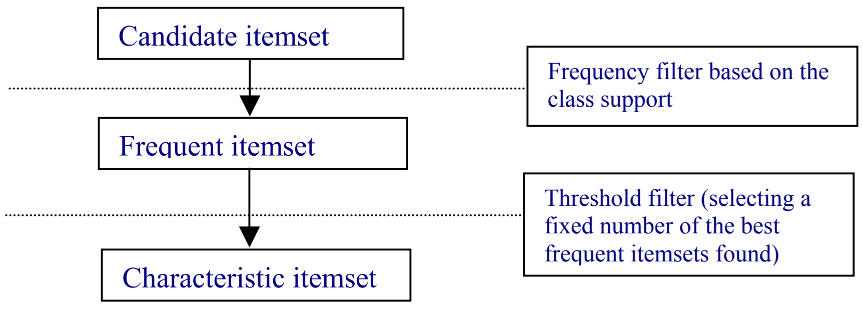
[18] This is, of course, an ad-hoc approach. I have tried various formulae leading to various results. It is likely that

Denominator is used to normalize with respect to the number of documents associated with topic T_i. It takes into account whether an itemset occurs in other topics as well. Significance of terms occurring frequently in documents other than DT_i is thus suppressed.

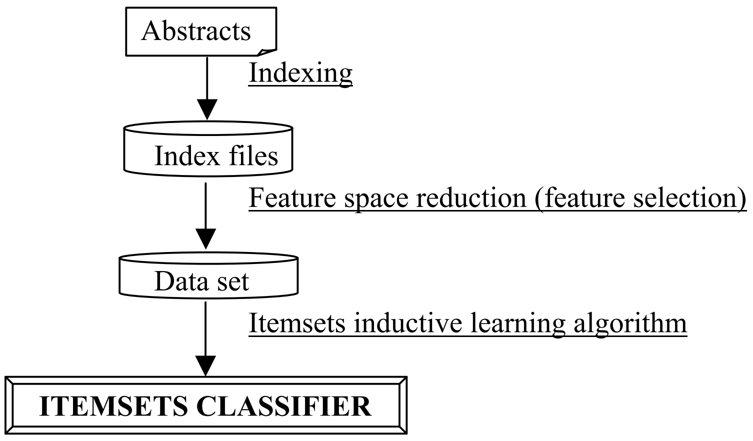
Upon associating itemsets with individual topics based on the formula above, we will acquire sets of itemsets C_i characterizing a particular topic[19] T_i. On the whole, there will be L sets of itemsets.

Aside: As we look for itemsets of higher cardinality, we are, in a way, performing *latent semantic indexing* (although applied to classifier training phase rather than document indexing). Co-occurring terms are projected onto the same dimension, while non-co-occurring terms onto different dimensions (latent semantic indexing results in dimensionality reduction). LSI is sometimes useful for solving vocabulary problem, at least partially. It is important, as objects in the collection of abstracts are often created by many different authors using various synonyms. *End of aside*

Let's summarize notation used herein: *itemset* is a set of items (i.e. terms). *Candidate itemset* is a potentially frequent itemset. *Frequent itemsets* are those appearing in more than *minsupport* number of documents of a given class. *Characteristic itemsets* are frequent itemsets declared characteristic for a given class (taking some fixed number of the best frequent itemsets associated with that class). See the figure below:


Candidate itemset
Frequency filter based on the class support
Frequent itemset
Threshold filter (selecting a fixed number of the best frequent itemsets found)
Characteristic itemset

Classifier construction is summarized in the figure below:


Abstracts
Indexing
Index files
Feature space reduction (feature selection)
Data set
Itemsets inductive learning algorithm
ITEMSETS CLASSIFIER

4.5.2. Classification Phase

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

I will come up with a different formula for the final version of this method.

[19] Each topic is currently represented by a set of itemsets of fixed size.

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

$$W_{T_i}^{D} = \sum_{j=1}^{|C_i|} wf_{|\Pi_j|} \times w_{\Pi_j}^{T_i} \quad where \;\; (\Pi_j \in C_i) \wedge (\Pi_j \subseteq \overline{D}) \; for \;\; all \; i = 1, 2, \ldots, L$$

In other words, the classification weight is determined by the sum of products of weights $w_{\Pi_j}^{T_i}$ with weight factors $wf_{|\Pi j|}$ for all itemsets of a given topic, which (the itemsets) are contained in the document being classified. Usage of $w_{\Pi_j}^{T_i}$ results in emphasizing those itemsets that provide the best description of topic T_i.

The document D will be associated with topic T_i corresponding to the highest weight W_{Ti}^{D}. Naturally, we can desire to associate the document with several topics. Should it be the case, we will classify the document D to all topics T_i where W_{Ti}^{D} exceeds a certain threshold value. Decreasing this threshold value may (but does not have to) result in lower precision (P) and higher recall (R) of classification. Modification of the threshold value generally leads to an opposite shift of precision and recall values.

4.6. Preliminary Results

Initial results achieved on a large collection of abstracts in English are quire promising. Results indicated in [32] were used for cross-method comparison, utilizing Reuters-21578 collection for this purpose. Classifiers in the test were used to categorize 12,902 documents into 118 classes, showing results for 10 largest categories, which include almost 75 % of all documents. That is why I have run the tests only for classes containing more than 180 documents (ranging from 212 to 2,779), to imitate approach of other authors. Documents were split to training and testing using 3:1 ratio.

Classifier	**Itemsets**	Naïve Bayes	BayesNets	Linear SVM
AVG(P, R)	**91.36**[20]	81.50	85.00	92.00

Preliminary results are motivating for further optimization of *itemsets* classifier, namely on other document collections, potentially in other languages, with the ultimate goal of developing a domain/language independent short-document classifier.

[20] When testing the classifier on training documents only, the average of P and R is 91.9 %.

5. AUTOMATIC DOCUMENT SUMMARIZATION

5.1. Introduction

Research in automatic document summarization is motivated by the need of applying *itemsets* classifier to full-length documents. As itemsets method is designed for short documents and abstracts, it is imperative to reduce the length of documents in the document collection by means of an intelligent summarizer. Documents "abbreviated" by automatic summarizer can be then passed to classifier to proceed with categorization.

Automatic document summarization also constitutes a *classification problem*: each sentence (or paragraph) in a document to be summarized is either a summary sentence (paragraph) or a non-summary sentence (paragraph). Class collection is restricted to two categories in this case.

It is imperative to define basic summary types:

a) *Indicative:* Indicative summaries give brief information on the central topic of a document (useful in IR applications, such as giving an indicative summary of documents retrieved by a search engine). Indicative summary is typically 5-10 % of the original text.

b) *Informative (substantive):* Informative summaries provide a substitute for full document ("surrogate", "digest"), retaining important details, while reducing information volume. Informative summary is typically 20-30 % of the original text.

c) *Evaluative:* Evaluative summary captures the point of view of the author on a given subject.

In the context of automatic document classification by itemsets method, we are interested primarily in *indicative* summarization (indicative summarizer has the ability to preserve the critical portion of the content); however, for automating the task of document summarizing in a digital library, *informative* summaries are needed, in order to maintain coverage of a topic.

5.2. Approaches to Document Summarization

Summary is either *fixed-length* (limited by some portion of document length, say 10 %) or *best-length* (no length limit is applied). Fixed-length summary will be preferred for the purpose of subsequent document categorization by itemsets method. *Optimum length* of documents being fed to itemsets-classifier is still a question of further research.

The following methodologies are most commonly used by document summarizers:

- Sentence length cutoff[21];
- Cue phrases[22];
- Sentence position in a document / paragraph;
- Occurrence of frequent words;
- Relative position of frequent terms within a sentence;
- Words in capital letters (uppercase words)[23];
- Occurrence of title words[24];
- Author-supplied abstract[25];
- Intra-document links between passages of a document.

The importance of term frequency for document summarization has been recognized by Luhn [26] as early as in 1958. Luhn observes that *relative position* of frequent terms within a sentence also furnishes useful measurement for determining significance of sentences. Significance of sentences can

[21] Short sentences tend not to be included in summaries.

[22] Summary includes sentences containing any of cue phrases, such as „in conclusion", „this letter", „as a result", „in summary", „to sum up", „the point is", etc.

[23] Proper names, abbreviations and acronyms are often important, increasing score of a corresponding sentence. Special attention must be paid not to include abbreviations of units of measurement (Kg, MPa, F, C, etc.)

[24] 80 % of significant words occurring in the title correspond to the most frequent significant words in the document [22].

[25] If an author-supplied abstract is present (heading containing the word *abstract*), subsequent paragraphs are used directly as the summary.

therefore be expressed as a *combination* of word *frequency* and *position* of these words. Intelligent summarizer should take into account linguistic implications, such as grammar, syntax, and possibly logical and semantic relationships. Speaking strictly of word frequency and word position, wherever the greatest number of frequently occurring different words are found in the greatest physical proximity to each other, the probability is very high that the information being conveyed is most representative of the document [26]. It is important to set a limit for the *distance* at which any two significant words will be considered *significantly related*. We have coped with a similar problem while implementing sliding-window modification (see Hynek, Ježek [3]) of the itemsets classifier.

Summary by extraction is mentioned by Kupiec, Pedersen and Chen [27]. The goal is to find a subset of the document that is *indicative* of its contents (sentences are scored and those with the best score are presented in a summary). It is important to note that extracted sentences rarely maintain narrative coherence of the original text. Sentence extraction is treated as a *statistical classification problem* in [27]. Classification function is developed in order to estimate the probability a given sentence is included in an extract. A training set of data (i.e. corpus of documents with labeled extracts) must be prepared manually prior to inductive learning process. A set of potential features, the classification method and a training corpus of document/extract pairs must be established for this purpose. Simple Bayesian classification function has been developed by Kupiec et al. [27] in order to assign for each sentence a score, which can be used to select sentences for inclusion in a generated summary. Resulting summaries are mainly indicative (give brief information on the central topic), with the average length of three sentences.

Strzalkowski et al. [28] observed that much of the written text display certain regularities of organization and style, which they call *Discourse Macro Structure* (DMS). Summaries are created to reflect the components of a given DMS. Resulting summaries are coherent and readable. DMS-based summarizer can generate both short indicative abstracts and well as longer informative digests that can serve as surrogates for the original text. In order to make a summary intelligible, it is necessary to extract text sections longer than simple sentences. Some studies [29] show that simply selecting the first paragraph from a document tends to produce better summaries than a sentence-based algorithm. Strzalkowski at al. [28] work on paragraph-level instead of sentences. Summaries are made up of paragraphs extracted from the original text. Indicative summaries are scored for relevance to pre-selected topics and compared to the classification of respective full documents. A summary is considered successful if it preserves the original document's relevance or non-relevance to a topic. By analogy, we can use the same evaluation method upon classifying original documents and their summarized counterparts by the itemsets classifier.

An interesting approach to document summarization is presented by Salton et al. [30], generating *intra-document links* between passages of a document, using these linkages to characterize the structure of the text. The knowledge of text structure is applied to perform automatic text summarization by passage extraction. Intra-document links are generated by means of techniques used by most automatic hypertext link generation algorithms. Needed to note that semantic links between documents are used by document clustering algorithms as well (see Section 6.1.). A text relationship map obtained by intra-document text linking may be used to isolate text passages that are functionally homogeneous [30]. These text passages represent contiguous piece of text that is well linked internally, but largely disconnected from the adjacent text.

5.3. Evaluation of Summarization Systems

Quality of an automatic summarizer can be measured by comparing *classification* of the original (full-length) document with that of the summarized document. Classification can be compared, for example, in terms of *precision*, *recall* or *F-measure*.

Besides evaluation based on classification, we may compare automatically generated extracts with those produced by humans. We must assume that a human would be able to identify effectively the most important sentences or paragraphs in a document. If the set of sentences/paragraphs selected by an automatic extraction method has a high overlap with the extract generated by human, the automatic summarizer should be regarded as effective. However, there is fairly uniform acceptance of the belief that any number of acceptable abstracts could effectively represent the content of a single document [31]. The essence of an idea can be captured by more than one sentence or phrase.

Evaluation of text summarization systems is discussed in detail by Firmin and Chrzanowski in [31]. Besides a number of other evaluation approaches, they mention degree of *domain independence*. Although most authors claim some degree of domain independence, they have performed tests only on a specific type of data, such as newspaper articles [29].

Quality of a summarizer was calculated, for example, as percentage of sentence matches and partial matches between their automatic summary and manually generated abstract [27]. The problem with this approach is reliance on the notion of a single "correct" ("best") abstract.

Quality of an abstract can be also measured by time required to read it. Firmin and Chrzanowski [31] compare average time required for reading full-text documents, best summaries, and 10 % summaries.

There are many improvements that can be made to the quality of the summaries, such as higher cohesion in sentence selection or sentence generation, and topic coverage across the set of topics mentioned within a document.

6. OTHER APPLICATIONS OF ITEMSETS METHOD

The following sections describe potential usage areas of the itemsets method. Further research will be performed to explore additional applications and new possibilities. Final results of practical implementation will be published.

6.1. Itemsets-Based Document Clustering

6.1.1. General Concepts of Document Clustering

The issue of clustering is closely related to classification. I will focus on document clustering based on textual contents of these entities. The goal of clustering is to maximize intra-class similarity while minimizing inter-class similarity. Clustering thus facilitates taxonomy building, i.e. information structuring into classes or a hierarchy of classes represented by similar entities. Clustering technologies are described in detail by Lukasová and Šarmanová [39] – clustering methods are divided into hierarchical (with further subdivision to agglomerative and divisive) and non-hierarchical (either optimizing or mode analysis methods).

Inter-document similarity can be expressed by a coefficient, so that we can associate each document with at least one cluster. Should similarity exceed some threshold value, documents rank into the same class of equivalence (cluster). Each new document can be compared, for example, with (a) the first document of a cluster (b) an arbitrary document of a cluster (c) a representative document of a cluster (d) several documents from a cluster, etc.

Clustering can be implemented either as hierarchical (creating a tree[26] of document clusters) or non-hierarchical. We will consider non-hierarchical clustering only. If we desire to join clusters into a hierarchical structure, we can apply various binding methods, such as “simple binding” (i.e. linking two most similar documents, each coming from a different cluster). By analogy, we can apply “full binding”, comparing two clusters using two least similar documents.

Clustering technology is used, for example, by Altavista search engine for clustering of results. Query results are substituted by a single document representing the whole web site. Such a document can be expanded to all relevant documents by clicking on “More pages from this site“ button.

6.1.2. Document Clustering Using Itemsets

In order to enhance the original use of *itemsets* classification method, we can also consider its application to document clustering. The following paragraphs contain preliminary ideas, while practical application is the issue of my further research.

We can start document clustering process with an arbitrary document clustering method, such as *k-means* or its modification, and create a variable number of clusters using pre-defined number of documents (at least one thousand). The number of clusters thus created depends largely on threshold values applicable to clustering method chosen for this purpose (genetic algorithm can be applied at this phase).

Let’s denote clusters containing at least, say, fifty documents, as *regular* clusters. Smaller clusters will be denoted as *non-regular*. We will use regular clusters for training the itemsets classifier, i.e. for creating characteristic itemsets for each of these clusters. The same number of characteristic itemsets will be defined for every regular cluster. This number can range from 10 to 40, for example, so that we could guarantee existence of enough characteristic itemsets for each regular cluster.

We will retrain itemsets classifier on regular clusters every time the number of documents in a particular cluster grows by a pre-defined figure (by applying a trigger). Itemsets classifier is retrained for a particular cluster only, as remaining clusters are still either non-regular, or trained enough.

New documents are assigned to non-regular clusters using a traditional document clustering method (such as *k-means*), until these clusters become regular. Itemsets classifier must be trained on each new regular cluster. Time requirements should not be prohibitive, as we look for frequent (and thus characteristic) itemsets only within the pertinent topic, not the whole document collection.

[26] By defining a tree, we create a concept hierarchy. A digital library can be treated (not necessarily) as several trees, i.e. several concept hierarchies.

“Cannot classify” criterion: This criterion is a must to decide whether we can classify a particular document at all. Let’s define it as: “computed weight of document assignment to the worst class compared to weight applicable to the best class is greater than X %”. Relatively narrow “weight band” means small difference between document ass;ignments to various topics. An assignment in such a case is too fuzzy. Classifier decides: cannot classify. It is to be decided what “narrow weight band” means. We may alter the width of this band depending on the number of topics resulting from document clustering process. We may apply some form of genetic algorithm for this purpose.

The first implementation draft of a clustering algorithm is depicted in the figure below.

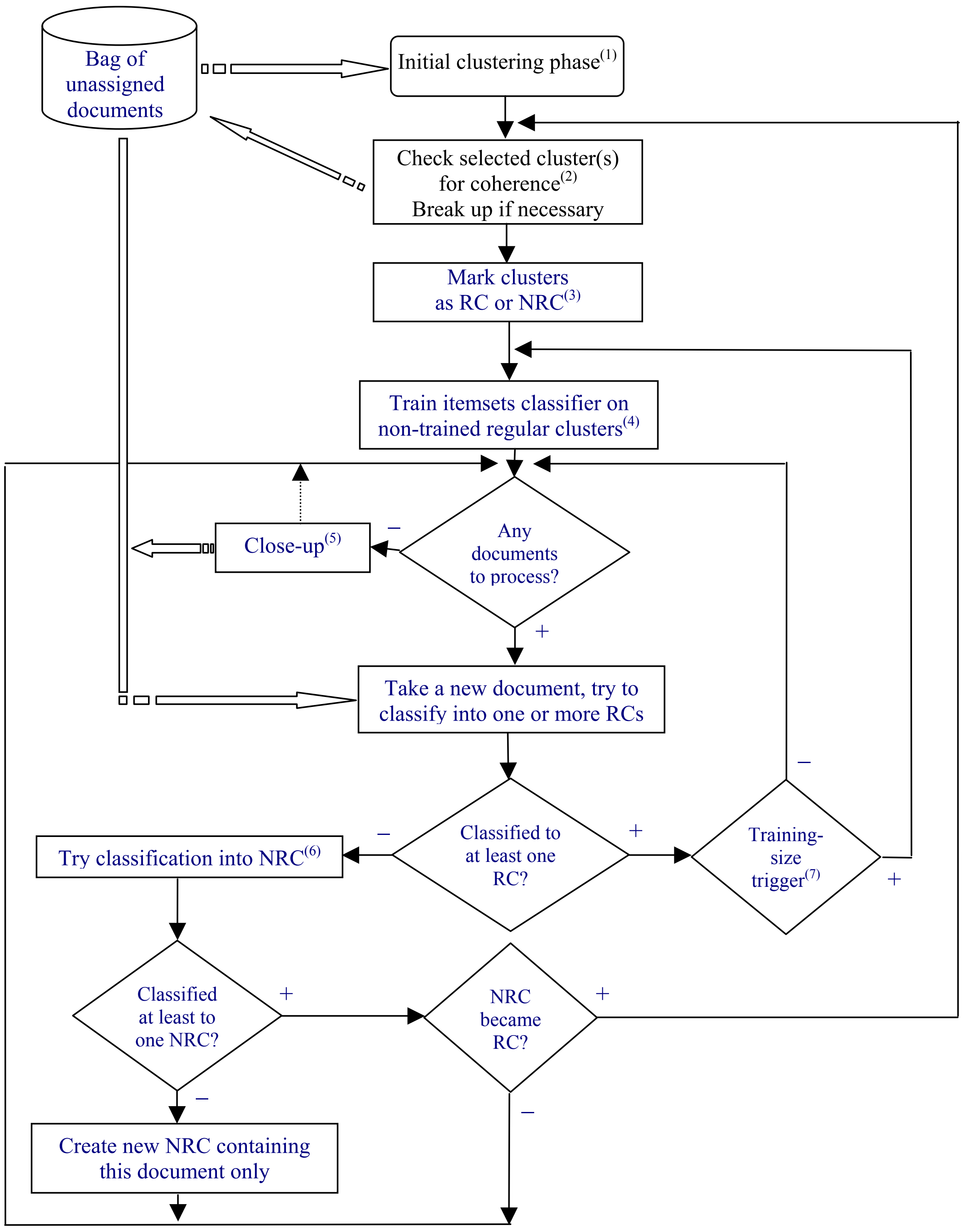

Bag of unassigned documents
Initial clustering phase(1)
Check selected cluster(s) for coherence(2) Break up if necessary
Mark clusters as RC or NRC(3)
Train itemsets classifier on non-trained regular clusters(4)
Any documents to process?
Close-up(5)
–
+
Take a new document, try to classify into one or more RCs
Classified to at least one RC?
–
+
Training-size trigger(7)
–
+
Try classification into NRC(6)
Classified at least to one NRC?
+
–
NRC became RC?
+
–
Create new NRC containing this document only

Fig. 6.1.2.-1: *A new concept of document clustering.*

Notes:

(1) Initial clustering phase is performed by means of a modified "traditional" clustering mechanism, such as *k-means* using itemsets in lieu of single terms. We will first evaluate the whole corpus by some form of *tf*idf* method, leaving out terms with extremely low or high frequency of occurrence, as well as applying stop-list and stemmer. We can then represent each document by several itemsets (achieving significant feature-space reduction).

(2) We need to ensure that one cluster does not contain semantically different documents. We can compare each pair of documents in the cluster. In case of inconsistencies, we must break up the cluster and put documents back into the bag.

(3) Regular clusters (RC) and non-regular clusters (NRC), depending on the number of documents assigned to the cluster.

(4) Clusters that were marked by training-size triggers are also considered non-trained regular clusters.

(5) Close-up operations: Browse cluster by cluster and look for similarity coefficients between each pair of documents (alike in step (2)). In the case of unsatisfactory semantic cohesion (i.e. the range of similarity coefficient is relatively too wide), break-up the cluster, and put documents back into the bag of unassigned documents.

Identify extremely small clusters, say of 1 or 2 documents. Put these back into the bag and change threshold values applicable to assignment.

Run clustering for remaining documents in the bag.

Optional: Take each document from every existing (both regular and non-regular) cluster and try to reclassify it (using *itemsets* classifier) into other regular clusters. We will thus ensure document classification into those clusters that may have been created after document's initial classification. This is the final phase of our clustering algorithm.

(6) Using the same clustering method as in the initial phase (e.g. modified *k-means*).

(7) The trigger is set ON if the cluster must be re-trained, i.e. more than a specified number of documents have been added since the last training.

The above approach ensures creating new clusters on as-needed basis, as well as break up of clusters demonstrating low semantic cohesion.

<u>Threshold values</u>: By properly setting up clustering threshold values, we can tune up optimal number of clusters upon the initial phase, or the total number of clusters in general. Trial and error approach (a form of genetic algorithm) is viable in this case.

As an alternative, we may try to assign a new document into one class only, which results in a simpler clustering method.

In the sense of taxonomy presented by Lukasová and Šarmanová [39], the clustering algorithm proposed above ranks to the class of *agglomerative* algorithms, although we are not building a hierarchy of documents (even if we could redesign the algorithm to a hierarchical one).

6.2. Naïve Bayes Classifier and its Modification Based on Itemsets Method

6.2.1. General Principles

Naïve Bayes classifier can be applied in such tasks where each instance x is described by a product of attribute values (probabilities), and target function $f(x)$ represents mapping to a finite set of values V (classes). If we describe the new instance A by n-tuple $(a_1, a_2 .... ,a_n)$, we can describe the target value of v_{MAP} as:

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j \mid a_1, a_2 .....a_n)$$

Then, by applying Bayes theorem:

$$v_{MAP} = \arg\max_{v_j \in V} \frac{P(a_1, a_{2,} ..... a_n \mid v_j) \cdot P(v_j)}{P(a_1, a_{2,} ..... a_n)}$$
$$= \arg\max_{v_j \in V} P(a_1, a_{2,} ..... a_n \mid v_j) \cdot P(v_j)$$

We can make an estimate of the parameters occurring in the above formulae:

- *$P(v_j)$* – based on frequency of v_j values in training data;
- *$P(a_1,a_2,....a_n \mid v_j)$* – these values can be estimated for sufficiently large sets of training data.

The Naïve Bayes classifier is based on a simplifying assumption of conditional dependence of attribute values of the target value. In other words, conjunction is represented by a product of probabilities of individual attribute values, i.e.: $P(a_1, a_2, ... a_n \mid v_j) = \prod_i P(a_i \mid v_j)$

By substituting to the above formula, we get **Naïve Bayes classifier**:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

Itemsets Modification

Instead of working with word attributes (a_i), we may use (characteristic) itemsets C of different cardinality for computing v_{NB}. Over the course of classification, we will utilize *characteristic* itemsets that have been determined for each class. This leads to significant feature space reduction. Only characteristic itemsets of a given class contained in the instance A being classified are utilized. The formula is therefore changed to

$v_{NBI} = \arg\max_{v_j \in V} P(v_j) \prod_i P(C_i^{v_j} \mid v_j)$, where $C_i^{v_j}$ represents i-th characteristic itemset of the class v_j, that also occurs in the instance A being classified. An itemset C was declared frequent in the class v iff its $P(C \mid v)$ exceeded some *minsupport* value. For filtering *characteristic* itemsets out of *frequent* ones, we are utilizing the same concept as in the itemsets classifier (see section 4.5.1.).

We may as well combine both methods, i.e. sum up the weights determined by NB and those determined by itemsets' modification of NB, designating this method **NBCI** (Naive Bayes Combined with Itemsets).

6.2.2. Document Classification Using Naive Bayes classifier

In order to illustrate practical use of Naive Bayes classifier, we will apply it to document classification. Let's have a document collection X, a set of training documents determining yet unknown target function *f(x)* and a finite set *V* of possible resulting values of the target function. Attributes $a_1, a_2....a_n$ represent individual words occurring in documents.

For the sake of simplicity, let's assume a set of 1,000 documents, 700 representing „interesting" documents, considering the remaining 300 documents uninteresting. The set *V* therefore represents two values only, $v_j \in \{interesting, uninteresting\}$. We can take a new document containing 111 words, to be classified into one of these classes, containing the word „your" in the 1st position, the word „letter" in the 2nd position, etc., and the word „date" in the last position. Now we apply Naive Bayes classifier:

$$v_{NB} = \arg\max_{v_j \in \{\text{interesting,uninteresting}\}} P(v_j) \cdot \prod_1^{111} P(a_i \mid v_j)$$

$$= \arg\max_{v_j \in \{\text{interesting,uninteresting}\}} P(v_j) \cdot P(a_1 = "your" \mid v_j) \cdot P(a_2 = "letter" \mid v_j) \cdots P(a_{111} = "date" \mid v_j)$$

In the end, we will get the following results:

v_{NB} ...classification via NB classifier

$v_{NBI}^{i_k}$ ...classification using characteristic *k-itemsets* (k being 1, 2, 3)

(not necessarily one class, can represent a set of classes)

The basic assumption we are making is mutual independence among words occurring at various positions in the document. Now we must determine probabilities of $P(v_j)$ and $P(a_i = w_k|v_j)$. We assume that $P(a_1 = w_k|v_j)$, $P(a_2 = w_k|v_j)$, etc. are equal to probability $P(w_k|v_j)$, where w_k is k-th word in vocabulary compiled from training documents.

$$\text{P(interesting)} = \frac{\text{interesting}}{\text{all}} = 0.7$$

$$\text{P(uninteresting)} = \frac{\text{uninteresting}}{\text{all}} = 0.3$$

In order to determine the probability of *P(w$_k$|v$_j$)* we will use formula (see [21]) for estimating probability of terms in document collection, i.e.

$$P(w_k \mid v_j) = \frac{n_k + 1}{n_j + |vocabulary|}$$

where |*vocabulary*| is the total number of distinct significant terms in the collection of training data
By analogy for itemsets modification: the total number of distinct frequent *itemsets* in the collection

n_j	– The total number of word positions in all training instances with the target value of v_j.
n_k	– The number of w_k occurrences over these word positions.

By analogy, the value of $P(C_i^{v_j} \mid v_j)$ can be determined by algorithm *classify (F,A)* described in [24], potentially also applying the weight factor of $w_{\Pi_j}^{T_i}$ specified in section 4.5.1. above.

6.2.3. Implementation of Naive Bayes document classification algorithm

Practical implementation is divided into training and classification phases.

<u>*Training phase*</u>

Input: Collection of training documents; Set of target values (categories, topics) *V*, where $v_j \in V$

Output: Files containing probabilities of *P(v$_j$)* and *P(w$_k$|v$_j$)*

Procedure

1. Identify all significant terms (other than stop words) contained in collection of training documents (*examples*)
   - *Vocabulary* ← set of all distinct words from the collection of training documents

2. Compute required probabilities of *P(v$_j$)* and *P(w$_k$|v$_j$)*
   - *docs$_j$* ← subset of documents from training collection with target value of v_j
   - *P(v$_j$)* ← $\frac{|docs_j|}{|examples|}$
   - *Text$_j$* ← one document formed by chaining all members of *docs$_j$*
   - n^j ← total number of different word positions in *Text$_j$*
   - For each word w_k from *Vocabulary*
     - n^j_k ← number of w_k occurrences in *Text$_j$*
     - *P(w$_k$|v$_j$)* ← $\frac{n_k^j + 1}{n_j + |vocabulary|}$

<u>*Classification phase*</u>

Input: Document being classified (*Doc*), a_i representing word occurring at i-th position

Output: The target value, i.e. category the document is classified to

- *position* ← all positions of words in *Doc* occurring in *Vocabulary*
- Compute value of v_{NB}, where $v_{NB} = \underset{v_j \in V}{\arg\max} P(v_j) \prod_{i \in position} P(a_i \mid v_j)$

Practical implementation of NBCI classification algorithm is currently an issue of further research.

Modification of Naive Bayes classifier with respect to itemsets is also described by Meretakis and Wüthrich [24]. *Large Bayes classifier* proposed Meretakis and Wüthrich is reduced to Naïve Bayes classifier when all itemsets are of size one only (i.e. no feature space reduction takes place). Support of itemsets is determined with respect to their occurrence in the whole document collection (using F as a global set of all interesting and frequent itemsets), not a particular class (my suggestion). They work with the largest possible itemsets, leaving out shorter itemsets contained in larger ones. In my opinion, this idea does not work well in practice.

My draft if based on the concept of *characteristic itemsets*, likely with a fixed number of characteristic itemsets for each class. Feature space dimension is reduced significantly in all cases, leading to higher speed and lower memory requirements. Some combination with weight factors determined in itemsets classifier might also improve classification results.

6.3. Itemsets-Based Information Search and Retrieval

6.3.1. Discovering Interesting Web Sites

We can use itemsets-based classifier for looking for potentially interesting web pages. Classification engine can run in a background, trying to classify candidate web pages into topics of interest pre-defined by the user (thus representing user's profile). Threshold values must be properly set to prevent information overload. Short average length of web pages makes practical implementation of this idea quite interesting.

Classifier can be trained on a directory containing web pages downloaded by the user and designated as *interesting*. It is also possible for the classifier to browse through "Favorites" stored by the web browser, visit all pages in each category, get trained (i.e. create sets of characteristic itemsets) and then look for additional pages that may fit into the corresponding category. Classifier could work in the background and notify the user of other potentially interesting web sites. Relevance ranking must be applied to prevent too many pages being offered to the user. Practical results of such a tool could be improved significantly by attaching user's rating of each web site visited by the user.

Output of the tool proposed herein would be represented by a web page containing links to recommended interesting web sites, sorted per relevance (a special web page per category in "favorites").

The tool could as well run permanently in the background while browsing on the Internet, highlighting HTML links recommended to the user, i.e. recommending web sites deemed interesting based on previous training on favorite web pages.

6.3.2. Information Push

The classifier mentioned in 6.3.1. could be as well trained on a single area of interest, boiling down to a simple binary classifier (*interesting/uninteresting*). Applying such a classifier on a scientific portal or a research digital library, we could filter out articles of user's interest only, pushing these to the user by e-mail. Similar concept can be applied to the digital library described in detail in Chapter 2 – *InfoServis* articles considered relevant based on user's profile can be distributed to users by e-mail.

6.3.3. Searching for Similar Documents

We can also use itemsets classifier for finding similar documents. Let's have document D with the task to find similar documents in our collection. We will threat this document as an item to be classified using itemsets method. We will therefore find classes $c_i...c_j$ (possibly one class only) based on characteristic itemsets occurring jointly in D and c_i ... c_j. We are then looking for "candidate similar documents", i.e. we are looking for documents from $c_i...c_j$, containing itemsets occurring also in D. In the case we find any documents, we sort them according to pre-defined weight criteria and provide these to the user.

The above is based on hypothesis that similar documents can be found in the same semantic topic. Classification threshold must be sufficiently "liberal", so that $c_i...c_j$ represent several classes, since documents can be classified into several topics, therefore similar documents can be stored in similar topics.

6.3.4. Querying

Let's create all possible k-itemsets from query *Q* entered by the user. These itemsets are then matched against characteristic itemsets associated with corresponding categories in document collection. If we find a match with a class, documents contained in this class are given higher priority for checking against *Q* – these documents will be assigned higher relevance. Maximum relevance corresponds to "match" over the largest *k-itemset*.

6.4. Topical Summarizer

The issue of summarization has been discussed in detail in Section 5. It is likely that itemsets-based approach could be also used for automatic document summarization.

Let's suppose we have a trained itemsets classifier, i.e. we have a set of characteristic itemsets for each class of document collection. We can regard all characteristic itemsets as phrases for a topical summarizer. We also have a document to be summarized. We can make a sophisticated match of this document against characteristic itemsets of every single class. The class matching the highest number of summary sentences is the one to which document should be classified (elementary principle of itemsets classification). Topical summary is a by-product, consisting of summary sentences containing characteristic itemsets of that particular class. Proper thresholds must be defined for selecting sentences as summary or non-summary.

6.5. Itemsets Spam Filter

6.5.1. Spam Filtering

The first spam filters were based on a simple principle – delete all messages containing pre-defined keywords. Obviously, such an approach leads to deleting many important messages at the higher than acceptable rate.

Modern spam filter is in fact a classifier trained on a single topic. By being classified into the "unsolicited mail" class, the incoming document is labeled as a spam; it is treated as a relevant message otherwise. Based on figures achieved by applying itemsets classifier on Reuters collection containing one topic only (nearly 100 % precision and recall), we may anticipate successful filtering results when applying our classifier on incoming e-mail messages. We can train the classifier on a representative collection of unsolicited messages. Such a collection can be created ad hoc, or one of numerous existing spam collections can be used.

We can use the table below to assess quality of itemsets-based spam filter:

Identified as a spam?	*Is it really a spam?*	
	Yes	No
Yes → Delete	a^{OK}	b^{error}
No → Keep	c^{error}	d^{OK}

Relevance factor applicable to **b** must be several orders of magnitude (e.g. λ = 1000) higher than that of **c** (i.e. we desire to prevent deleting important messages). In other words, blocking legitimate message is λ-times more costly than misclassifying a spam message.

Classification accuracy (i.e. filter accuracy) can be defined as $\frac{a+d}{a+b+c+d}$

Precision $P=\frac{a}{a+b}$, Recall $R=\frac{a}{a+c}$

6.5.2. Results of a Preliminary Practical Implementation

Itemsets spam filter was tested on an "encrypted" collection of legitimate and spam messages, dubbed PU1 corpus[27], which was made publicly available contributing to standard benchmarks. Also ten-fold

[27] PU1 corpus is publicly available from the publications section of http://www.iit.demokritos.gr/~ionandr

cross validation was introduced to reduce random variations and an effect of attribute-set size, stemming and stop-list was taken into account.

PU1 testing corpus[28] consists of 1099 messages: 481 spam messages in the English language, not containing duplicates; 618 legitimate messages. There are no more than five messages from each person because the reader usually saves the friend's address into the address book in the e-mail browser and in the future the messages from these people will not have to be examined.

Attachments and HTML tags were removed from all messages to respect privacy, in this publicly available version of PU1, fields other than "Subject:" were removed, and each token (word, number, punctuation mark, etc.) in the bodies or subjects of these messages was replaced by a unique number. For example:

```
From: spammer@spamcompany.com
To: spamtarget@provider.com
Subject: Get rich now !
Click here to get rich ! Try it now !
```

becomes:

```
Subject: 1 2 3 4
5 6 7 1 2 4 8 9 3 4
```

There are actually four "encrypted" versions of the publicly available of PU1 corpus, one for each combination of enabled/disabled stop-list and stemmer.

PU1 corpus statistics:

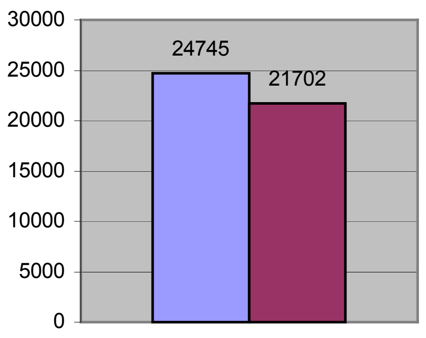
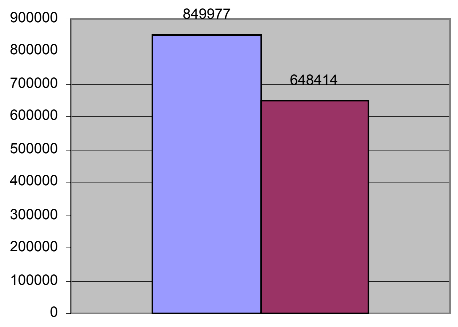
E-mail messages: 1099 (481 spam, 618 legitimate)

Classes: 2 (spam, legitimate)

Total length of e-mail messages: 849,977 terms

Number of unique significant terms: 24,745

Length of e-mail messages			
	Minimum	Maximum	Average
All terms	16	15677	773
Significant terms	12	14367	590

Impact of stemming:	*Impact of stop-list application:*
Number of unique significant terms after stemming: 21,702 (87.70 % of the original)	The total length of the e-mail messages upon applying a stop-list: 648,414 (76.29 % of the original)
	

Testing

In all experiments with *itemsets* classifier, *ten-fold cross validation* was used to reduce random variations. That is, PU1 was partitioned randomly into ten parts, and each experiment was repeated ten times, each time reserving a different part for testing, and using the remaining nine parts for training.

[28] For detailed information on PU1 corpus, see [23]

Results (applicable to 500 features) were then averaged over the ten runs.

Filter used	**(P+R) / 2 (%)**
(a) Itemsets	97.63
(b) Itemsets + stemmer	97.72
(c) Itemsets + stop-list	97.54
(d) Itemsets + stemmer and stop-list	97.81

We can also experiment with 2-itemsets, 3-itemsets, and 4-itemsets. In fact, this results in very little improvement, while increasing memory requirements and processing time enormously.

Implications

Practical implementation of an itemsets-based spam filter leads to very promising results. 2-itemsets, 3-itemsets, and 4-itemsets are not worth using because of significant time and memory requirements. Neither stemmer nor stop-list improved precision and recall of the filter significantly.

The following spam filters can be also considered for comparison purposes:

Spam Killer (www.spamkiller.com)

Anti Spam (securedisk.netfirms.com)

7. CONCLUSION

This report presents a survey of issues related to document classification, clustering, and summarization. Practical implementation of a digital library is also presented. A survey of current text categorization algorithms is provided, along with a preliminary idea of a new classification method, *itemsets classifier*, and its potential applications.

My further research will be focused namely on optimization and ramifications of *itemsets* document categorization, specifically the cross-collection and cross-method testing. Viability of its application on full-text document collections will be tested upon coming up with a suitable automatic document summarizer, which is an issue of yet another research.

Itemsets classifier will be optimized namely by improving the stemming algorithm, making detailed assessment of numerous input parameters and threshold values of the classifier, testing the impact of its sliding-window modification for various document collections, optimizing itemsets-generation phase for time and memory requirements.

My further study will be devoted to other applications of *itemsets* method, such as spam filtering, document clustering, and modification of Naïve Bayes classifier. Any promising results achieved herein will result in integrating the applicable method into the real world digital library in order to back-up results by real-life use. Special attention will be paid to cross-cultural issues in the context of digital libraries, i.e. significant differences between Czech and English document collections. Besides implementing these preliminary ideas, I will try to come up with even further applications of this promising method.

Itemsets classifier has already been subject to extensive testing (see author's publications below), providing quite promising results (namely in the case of Reuters-21578 collection), comparable to those generated by SVM categorization algorithm, in terms of precision, recall and storage/time requirements. Further work is needed to improve classification results on Czech documents, namely unabridged full-text files.

8. WEB INFORMATION SOURCES

Pair-wise Comparison	http://lsa.colorado.edu/cgi-bin/LSA-pairwise.html
Natural language querying	http://www.ask.com
Newsgroups	http://www.dejanews.com
Terminology	http://www.onelook.com
Search engines	http://www.searchenginewatch.com
Abstracts	http://www.reserse.cz
Computer Science Technical Reports (Digital library of the New Zealand)	http://www.nzdl.org/cstr
Clustering (CLUTO clustering toolkit)	http://www.cs.umn.edu/~karypis/cluto
Text databases	http://www.research.microsoft.com/research/db/debull
	http://www.informatik.uni-trier.de/~ley/db/groups.html
Digital Libraries	
D-Lib® Forum	http://www.dlib.org/
Virginia Tech Courseware (self-study course in digital libraries)	http://ei.cs.vt.edu/~dlib
ACM Digital Library	http://www.acm.org/dl
Knowledge-based projects	http://www.cs.utexas.edu/users/mfkb/related.html
Dublin Core	http://dublincore.org
Springer – Knowledge and Information Systems	http://link.springer.de/link/service/journals/10115/index.htm
Diglib - Implementation Issues	
MySQL	http://www.tcx.se
	http://www.mysql.com
PHP	http://www.php.cz
	http://www.php.net
	http://www.pruvodce.cz/kluby/php3
	http://www.phpbuilder.com
Apache	http://www.apache.org
	http://sunsite.mff.cuni.cz/web/apache
Text corpuses	
Reuters-21578	http://www.research.att.com/~lewis/reuters21578.html
Czech national corpus	http://uckn.ff.cuni.cz/CZ/cnc
Medline	http://www.nlm.nih.gov/databases/databases_medline.html
UCI Machine Learning Repository	http://www.cs.uci.edu/~mlearn/MLRepository.html

;

9. REFERENCES

[1] Houser P.: „Hledat na Internetu dnes ještě neznamená najít“, Technology World, Computerworld 1-2/99, pp. 4

[2] Kosek J.: „XML: Další magická zkratka internetového světa“, Computerworld 14/99, pp. 11

[3] Jonák Z.: „Zbraně proti entropii Internetu“, Technology World, Computerworld 7/99, pp. 4

[4] Hejný J.: „Tma pod svícnem aneb informace skryté za humny“, Technology World, Computerworld 6/99, pp. 2

[5] Čáp J.: „Katalogy vs. Indexy“, PC World 1/1999, pp. 130

[6] Date C.J.: “An Introduction to Database Systems”, Volume I, 5th Edition, Addison Wesley Publishing Company, 1990

[7] Pokorný J.: „Databázová abeceda“, Science 1998

[8] Pola M.: „Od klínového písma k fulltextu“, Chip 9/1996, pp. 44

[9] Vaňous J.: „Interaktivita na WWW“, Softwarové noviny 1/1997, pp. 88

[10] Kroha P.: „Databáze dnes a zítra“, Softwarové noviny 9/1995, pp. 18

[11] Kukal J.: „Znalostní databáze“, Softwarové noviny 9/1998, pp. 96

[12] Kostelanský J.: „Fuzzy SQL“, Computer Echo 1/95, pp. 14

[13] Pokorný J.: „Počítačové databáze“, Kancelářské stroje 1991

[14] Lake, Tweney, Li-Ron: „Najděte to na Webu“, PC World 9/1999, pp. 76

[15] Peterka J.: „Co může Internet nabídnout?“, Softwarové noviny 10/1997, pp. 30

[16] Kosek J.: „PHP Tvorba interaktivních internetových aplikací“, Grada 1999

[17] Pokorný J., Snášel V., Húsek D.: „Dokumentografické informační systémy“, Karolinum 1998

[18] Melichar B.: „Textové informační systémy“, Vydavatelství ČVUT 1996

[19] Psutka, Kepka: „Umělá inteligence – reprezentace znalostí“, Západočeská univerzita 1994

[20] Bulletin of the Technical Committee on Data Engineering, June 1998, Vol. 21, No. 2, IEEE Computer Society

[21] Mitchell T.M.: “Machine Learning”, McGraw Hill 1997

[22] Právník 137/98, ISSN 0231-6625, pp. 135-150

[23] Androutsopoulos I., Koutsias J., Chandrinos K.V. and Spyropoulos C.D.: “An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages“, Proceedings of the 23rd Annual International ACM SIGIR 2000, Athens, Greece, pp. 160-167

[24] Meretakis D., Wüthrich B.: “Extenting Naïve Bayes Classifiers Using Long Itemsets”, KDD-99, San Diego, California, 1999, pp. 165-174

[25] Agrawal et al.: „Advances in Knowledge Discovery and Data Mining“, MIT Press 1996, pp. 307-328

[26] Luhn H.P.: „The Automatic Creation of Literature Abstracts“, IRE National Convention, New York, March 24, 1958

[27] Kupiec J., Pedersen J., Chen F.: “A Trainable Document Summarizer”, Xerox Palo Alto Research Center, Palo Alto, CA 94304

[28] Strzalkowski T., Stein G., Wang J., Wise B.: “A Robust Practical Text Summarizer”, GE Corporate Research and Development, Niskayuna, NY 12309

[29] Brandow R., Mitze K., Rau L.: “Automatic Condensation of Electronic Publications by Sentence Selection”, Information Processing and Management 31(5), 1995, pp. 675-686

[30] Salton G., Singhal A., Mitra M., Buckley C.: “Automatic Text Structuring and Summarization”, Dept. of Computer Science, Cornell U., Ithaca NY

[31] Firmin T., Chrzanowski M.: “An Evaluation of Automatic Text Summarization Systems”, Department of Defense, Ft. Meade MD 20755

[32] Dumais S., Platt J., Heckerman D.: “Inductive Learning Algorithms and Representations for Text Categorization”, CIKM 98, Bethesda MD, USA, pp. 148-155

[33] Yang Y., Liu X.: “A Re-examination of Text Categorization Methods”, SIGIR 99, 8/99, Berkley, CA, USA, pp. 42-49

[34] Rijsbergen van C.J.: “Information Retrieval”, Butterworths, London, 1979

[35] Ortega J., Koppel M., Argamon S.: “Arbitrating Among Competing Classifiers Using Learned Referees”, Knowledge and Information Systems (2001) 3: 470-490

[36] Schapire R.: “The Strength of Weak Learnability”, Machine Learning 5(2): 197-227, 1990

[37] Haas S., Grams E.: “Page and Link Classifications: Connecting Diverse Resources”, Digital Libraries 98, Pittsburgh PA, USA, ACM 1998, pp. 99-107

[38] Beney J., Koster C.: „Classifying Large Document Collections with Winnow”, November 2000

[39] Lukasová A., Šarmanová J.: „Metody shlukové analýzy“, SNTL Praha, 1985

Author’s Conference Publications Related to this Report:

[1] Hynek J., Ježek K.: „Automatická klasifikace dokumentů do tříd metodou Itemsets, její modifikace a vyhodnocení“, proceedings of Datakon 2001, pp. 329-336, Brno, 20. – 23. 10. 2001, ISBN 80-227-1597-2

[2] Hynek J., Ježek K., Rohlík O.: „Short Document Categorization – Itemsets Method“, proceedings of PKDD 2000, Lyon – France, 12. – 16. 9. 2000

[3] Hynek J., Ježek K.: „Document Classification Using Itemsets“, proceedings of MOSIS 2000, pp. 97-102, Rožnov pod Radhoštěm, 1. – 4. 5. 2000, ISBN 80-85988-45-3

Other publications

- Hynek J., Vítkovský R.: Anglicko-český a česko-anglický slovník výpočetní techniky a informačních technologií, 520 pages, Fraus, March 2000
- Hynek J., Vítkovský R.: Anglicko-český a česko-anglický slovník výpočetní techniky a informačních technologií, 2nd Edition, 560 pages, Fraus, to be published in 2002

Translated books

- **AutoCAD 2002 – Podrobný průvodce**, George Omura, 1050 pages, Grada Publishing, to be published in 2002
- **Delphi 6 – příručka zkušeného programátora**, Marco Cantú, 650 pages, Grada Publishing, to be published in 2002
- **Delphi 6 – příručka programátora**, Marco Cantú, 640 pages, Grada Publishing, to be published in 2002
- **Myslíme v jazyku C#**, Tom Archer, 450 pages, Grada Publishing, 2002
- **Administrace systému Linux**, Steve Shah, 650 pages, Grada Publishing, 2002
- **Myslíme v jazyku UML**, Joseph Schmuller, 390 pages, Grada Publishing, 2001
- **Myslíme v jazyku C++**, Bruce Eckel, 556 pages, Grada Publishing, 2000
- **Outlook 2000**, Gini Courter, Annette Marquis, 568 pages, Grada Publishing, 2000
- **AutoCAD 2000 – Podrobný průvodce**, George Omura, 934 pages, Grada Publishing, 1999
- **Delphi 4 – Podrobný průvodce programátora**, Marco Cantú, 637 pages, Grada Publishing, 1999