ulm university universität
uulm

Novel Methods for Text Preprocessing and Classification

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

(DR.-ING.)

der Fakultät für Ingenieurwissenschaften
und Informatik der Universität Ulm

von

Tatiana Gasanova

aus Krasnojarsk (Russland)

Gutachter:	Prof. Dr. Dr.-Ing. Wolfgang Minker Prof. Dr. Eugene Semenkin Prof. Dr. Günther Palm
Amtierender Dekan:	Prof. Dr. Tina Seufert

Ulm, 3.Juli 2015

PhD thesis: Novel Methods for Text Preprocessing and Classification

Written text is a form of communication that represents language (speech) using signs and symbols. For a given language text depends on the same structures as speech (vocabulary, grammar and semantics) and the structured system of signs and symbols (formal alphabet). Written text has always been an instrument of exchanging information, recording history, spreading knowledge, maintaining financial accounts and formation of legal systems.

With the development of computers and Internet the amount of textual information in digital form has dramatically grown. There is an increasing need to automatically process this information for variety of tasks related to text processing such as information retrieval, machine translation, question answering, topic categorization and topic segmentation, sentiment analysis etc. Many important text processing tasks fall into the field of text classification. Text classification is a task of assigning one or more class labels for each document from a collection of textual documents. A set of class labels is usually predefined. In order to deal with textual data using existing classification algorithms, a textual document has to be preprocessed in a way that classification algorithms can handle. Methods that transform textual data into vectors of features are called text preprocessing methods. Words, phrases or clusters of synonyms are often considered as features or terms. Most of text preprocessing algorithms are based on an idea of term weighting, where for each feature/term a measure of "importance" is calculated. This term "importance" value is called a term weight and it depends on statistics of term occurrence in the document and in the whole collection. Term weighting algorithms can be unsupervised, if they are based only on statistical data of term occurrences and do not use information about class labels, and supervised, if they use such information.

This thesis addresses the development and evaluation of novel text preprocessing methods, which combine supervised and unsupervised learning models in order to reduce dimensionality of the feature space and improve the classification performance. Metaheuristic approaches for Support Vector Machine and Artificial Neural Network generation and parameters optimization are modified and applied for text classification and compared with other state-of-the-art methods using different text representations.

In general, the choice of text preprocessing methods makes a significant impact on the performance of classification algorithms. There are some techniques that combine supervised and unsupervised algorithms in order to take advantage of the much larger set of the unlabelled

documents. Another direct application of the unsupervised approach is the term clustering algorithm to the feature space reduction. Some semi-supervised methods use the unsupervised criterion to explore the relations between the objects from the same class and objects from different classes.

The methods developed in this thesis have been evaluated using six corpora written in English and French languages and designed for different tasks: document categorization (DEFT 2008 campaign), natural language call routing (SpeechCycle company) and opinion mining (DEFT 2007 campaign).

In this thesis a novel supervised term weighting method has been proposed. It has been compared with state-of-the-art term weighting techniques both supervised and unsupervised. Numerical experiments conducted across the variety of the classification algorithms (Naive Bayes, Support Vector Machine, Rocchio classifier, k-nearest neighbor algorithm and Artificial Neural Network) have shown that in terms of classification quality the proposed approach performs equally well or better than a ConfWeight (a popular supervised term weighting algorithm based on the evaluation of the confidence intervals) and outperforms TF-IDF (standard unsupervised term weighting). Moreover, the novel term weighting scheme can be calculated with the same computational time as TF-IDF approach, which is 3-4 times faster than the ConfWeight.

The method introduced in this thesis is based on the combination of the supervised and unsupervised learning models that improve the performance of the supervised classification algorithm by clustering the class with a non-uniform structure. In a call routing task there is often a category where all calls which do not have a meaning are routed. Examples of this type of calls are: if the call does not contain meaningful words; if the call is not related to any of the possible call reasons (there is a predefined set of call categories); if the call can be routed to many categories with the same probability. These cases form a relatively large class, which contains utterances with different meanings. Detection of this non-uniform class is a harder problem in comparison to the detection of other well defined classes. The method proposed in this work assumes that if this large class is divided into smaller subclasses then the classification performance will improve. Since there are no labels for these subclasses, some unsupervised technique should be applied. In this thesis the hierarchical agglomerative clustering has been used for a non-uniform class from the call routing task. After the class is divided into the set of clusters, the classification algorithm treats these clusters as normal classes among with the existing ones. This method improves the performance by up to 7%.

This thesis is also concerned with the dimensionality reduction of the feature space. The aim is to develop methods that can reduce the term set without the need of stop-word or ignore-word filtering and stemming since these methods are domain-related or require linguistic knowledge and therefore cannot be transported to other tasks directly. The approach

proposed in this work consists of term clustering and recalculation of the weights of the new terms (obtained term clusters). As a term clustering algorithm, the hierarchical agglomerative clustering with complete linkage has been chosen, since the output of this method is a hierarchical tree of terms which can be cut at the required length (the chosen number of term clusters).

Since the reduced term set is relatively small, an optimization method can be applied for term weights in order to improve classification results. An accuracy rate obtained on the training data is used as an optimization criterion (objective function). As the term weights are modified the criterion changes. Due to the lack of information about the objective function behaviour, heuristic algorithms such as genetic algorithms are usually a good choice. Since the task of optimizing the weights of each category can be naturally divided into a set of subtasks, which can be solved in parallel, coevolution should be applied. In this thesis a coevolutionary genetic algorithm with cooperative scheme is used.

Finally, a new representation of documents is calculated based on the obtained weights of term clusters. Experimental results conducted in this thesis have shown good performance in comparison with the results obtained by the participants of the DEFT 2007 and DEFT 2008 campaigns in case of Books, Games, Debates, T1 and T2 corpora; and the results obtained with the state-of-the-art classification methods in case of SpeechCycle corpus.

Contents

List of Figures

List of Tables

Publications

Parts of this thesis are based on the material published in the following papers.

- R. Sergienko, T. Gasanova, E. Semenkin and W. Minker, "Collectives of Term Weighting Methods for Natural Language Call Routing", Lecture Notes in Electrical Engineering, Informatics in Control Automation and Robotics - ICINCO 2014 (2016).
- T. Gasanova, R. Sergienko, E. Semenkin and W. Minker, "Dimension Reduction with Coevolutionary Genetic Algorithm for Text Classification", Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Vienna University of Technology, Austria, Vol. 1, pp. 215-222, September 2014
- R. Sergienko, T. Gasanova, E. Semenkin and W. Minker, "Text Categorization Methods Application for Natural Language Call Routing", Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics - ICINCO 2014, Vienna, Austria, Vol. 2, pp. 827-831, September 2014
- S. Akhmedova, E. Semenkin, T. Gasanova and W. Minker, "Co-operation of biology related algorithms for support vector machine automated design", International Conference on Engineering and Applied Sciences Optimization (OPT-i), Kos Island, Greece, June 2014
- T. Gasanova, R. Sergienko, S. Akhmedova, E. Semenkin and W. Minker, "Opinion Mining and Topic Categorization with Novel Term Weighting", Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Association for Computational Linguistics, Baltimore, Maryland, USA., pp. 84–89, June 2014
- T. Gasanova, R. Sergienko, W. Minker, E. Semenkin and E. Zhukov, "A Semi-supervised Approach for Natural Language Call Routing", Proceedings of the SIGDIAL 2013 Conference, pp. 344-348, August 2013

- R. Sergienko, W. Minker, E. Semenkin and T. Gasanova, "Call Routing Problem Solving Method Based on a New Term Relevance Estimation", Program Products and Systems, Num. 1, pp. 90-93, March 2013
- T. Gasanova, R. Sergienko, W. Minker and E. Zhukov, "A New Method for Natural Language Call Routing Problem Solving", Bulletin of Siberian State Aerospace University named after academician M.F. Reshetnev, Num. 4, 2013
- T. Gasanova, R. Sergienko, W. Minker and E. Semenkin, "Text Categorization by Coevolutionary Genetic Algorithm", Bulletin of Siberian State Aerospace University named after academician M.F. Reshetnev, Num. 4, pp. 104-108, 2013

Chapter 1

Introduction

1.1 Motivation

Nowadays, with the development of computers and World Wide Web, a great amount of textual data is being generated in digital form. There are a lot of actual applications related to automatic processing of such textual information including information retrieval, machine translation, question answering, topic categorization and topic segmentation, sentiment analysis etc. Many of these problems fall into the field of text classification.

The problem of text classification finds applications in a wide variety of domains in text mining that are discussed in detail in Aggarwal & Zhai (2012). Some examples of domains in which text classification is commonly used are as follows:

- **News filtering and Organization**
  Nowadays most of the news services are electronic and every day a large volume of news articles are created. In such cases, it is difficult to organize the news articles manually. Therefore, automated methods can be very useful for news categorization in a variety of web portals as discussed in Lang (1995). This application of text classification is also referred to as text filtering.

- **Document Organization and Retrieval**
  The text filtering can be generally used for many other applications. A lot of supervised methods may be applied for documents organization in many domains. These include large digital libraries of documents, web collections, scientific literature etc. Hierarchically organized document collections are particularly useful for browsing and document retrieval as discussed in Chakrabarti *et al.* (1997).

- **Opinion Mining**
  Customer reviews or opinions are often short text documents which can be searched

for the information useful for companies. Details on application of text classification methods in order to perform opinion mining are discussed in Liu & Zhang (2012).

- **Email Classification and Spam Filtering**
  It is often desirable to classify email as discussed in Carvalho & Cohen (2005); Cohen (1996); Lewis & Knowles (1997) according to the subject or to determine junk email as in Sahami *et al.* (1998) in an automated way. It is also referred to as spam filtering or email filtering.

In general, given a set of predefined class annotations (class labels) text classification is formulated as a task of assigning one or more class labels for each document from a collection of textual documents. These class labels indicate object types, in text classification case a document topic, an attitude of the writer or a problem of a caller to technical support. If there is available information about class labels of some documents, this data can be used to train a classification algorithm, i.e. to build a system that approximates the class labels of the given objects and is able to generalize to the objects with unknown labels. In order to deal with textual data using existing classification algorithms the textual document has to be preprocessed in a way that classification algorithms can handle, i.e. vector space model or term vector model. Methods that transform textual data into vectors of features are called text preprocessing methods. Words, phrases or clusters of synonyms are often considered as features or terms (bag-of-words model). Most of text preprocessing algorithms are based on an idea of term weighting, where for each feature/term a measure of "importance" is calculated. This term "importance" value is called a term weight and it depends on statistics of term occurrence in the document and in the whole collection. Term weighting algorithms can be unsupervised, if they are based only on statistical data of term occurrences and do not use information about class labels, and supervised, it they use such information.

The difference between the text classification task and other classification problems is the need of first transforming or preprocessing textual information in a form that classification algorithms are able to process. Figure 1.1 illustrates the basic scheme of text classification process. This thesis addresses the development and evaluation of novel text preprocessing and text classification methods. The task of classifying the text documents consists of text preprocessing and text classification steps. Text preprocessing algorithms transform the textual data into real-valued (or binary-valued) vectors. In the second stage classification methods are applied to the obtained vectors. Text preprocessing algorithms usually involve the choice of the feature space (or term set) that is made on the basis of statistical co-occurrence of words, manually created stop-word and ignore-word lists (these lists include words that are irrelevant to the given classification task such as prepositions, pronouns et cetera) or feature space reduction algorithms that usually use the domain specific information.

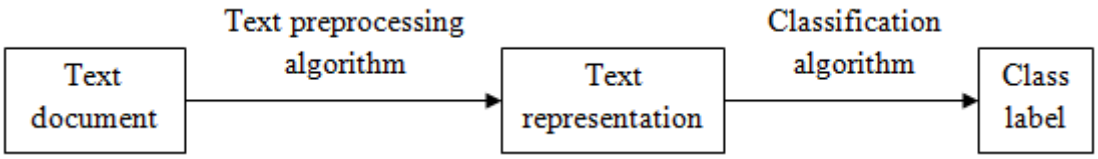

Figure 1.1: Text classification process

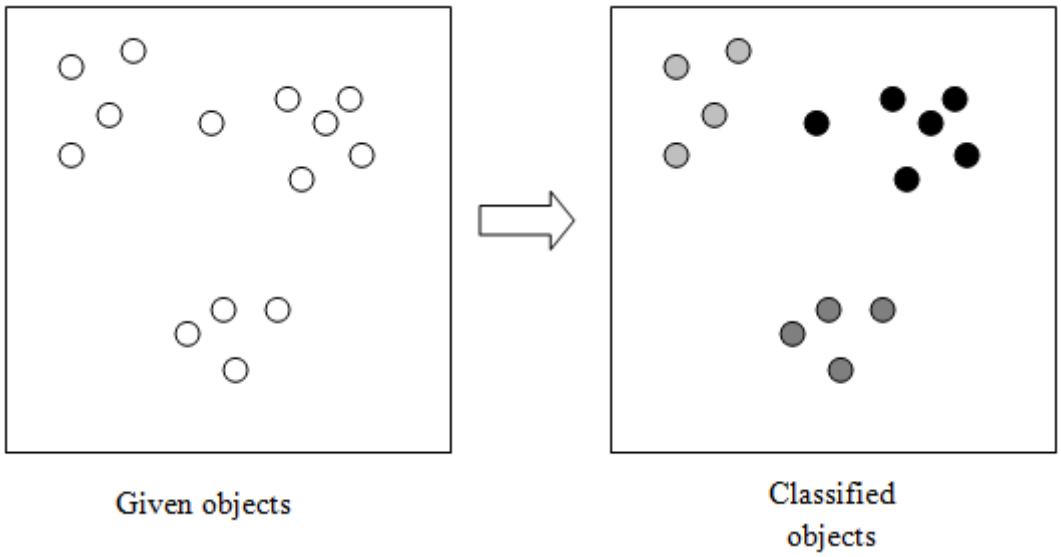

Figure 1.2: Unsupervised classification

The main objective of this thesis is the development of text preprocessing and text classification methods that do not require linguistic or domain related information such as manually chosen stop-word or ignore-word lists or any a-priori known morphological information. This thesis aims to investigate the text classification algorithms that can be easily transported to another domain or language without major changes. In order to address such objective, this thesis has been focused on two major fields: text preprocessing and text classification.

In general classification can be defined as a task of dividing the given objects into the set of predefined categories (object types). The classification task can be an unsupervised or a supervised one. Unsupervised classification (clustering) is where there is no available information about categories, no so-called train set and the task is to analyze the relations between the objects themselves and divide them according to some unsupervised criterion. Figure 1.2 shows an example of unsupervised classification, where only feature vectors of the given objects are known. A clustering algorithm analyzes these feature vectors and separates the objects in a way that the obtained clusters tend to be dense and far from each other.

In contrast, the supervised classification has a-priori information about the class labels of

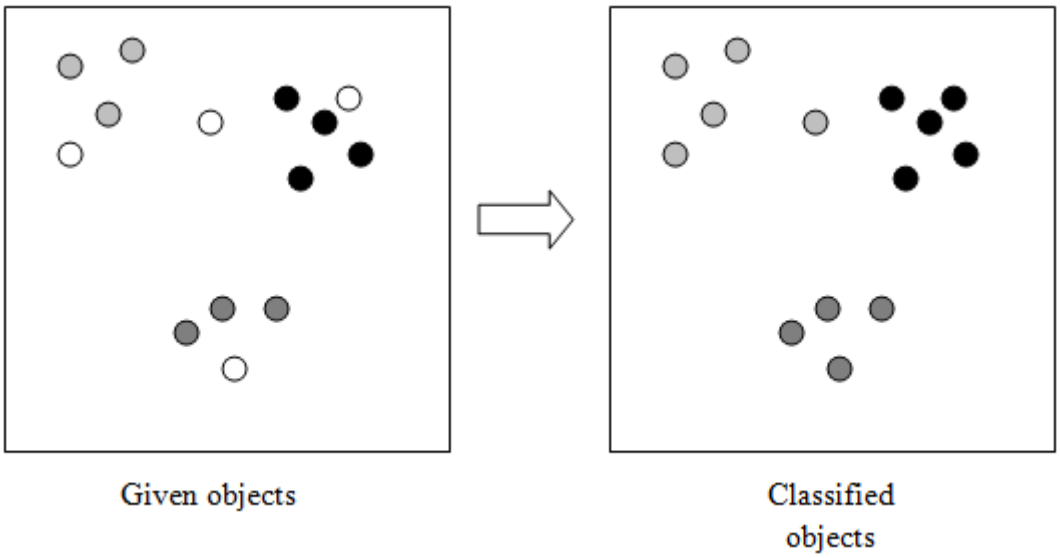

Figure 1.3: Supervised classification

some of the objects and for each of the test objects the task is to find the most similar group of objects. Figure 1.3 shows an example of supervised classification, where feature vectors of all the given objects are known as well as class labels of some objects. A classification algorithm analyzes this information and builds a system that predicts class labels of other objects with unknown labels.

Text classification consists of two main stages. The first one is the preprocessing stage where each document has to be represented in a way that classification algorithms are able to process. The most common representation is a vector space model that considers each document as a vector. What will be defined as coordinates in the feature space and how to transform the text document to the vector in this space are the tasks of the text preprocessing method. Usually text preprocessing methods include some ways to reduce the feature space dimensionality. The second stage of text classification is a general classification of the obtained vectors in the feature space.

Figure 1.4 shows an example of text classification. There are two text documents that belong to different classes: math and tale. The simplest document representation is called bag-of-words model where each vector length equals the size of dictionary and vector components indicate the number of times terms occur in this text document. However, in text classification tasks simpler binary features are often used, where the vector component indicates presence or absence of the term in this document. First, the documents from Figure 1.4 have been preprocessed into the binary form. In this example the feature space consists of all words appeared in both documents: “so”, “X”, “is”, “equal”, “to”, “Y”, “princess” and “cute”. Second,

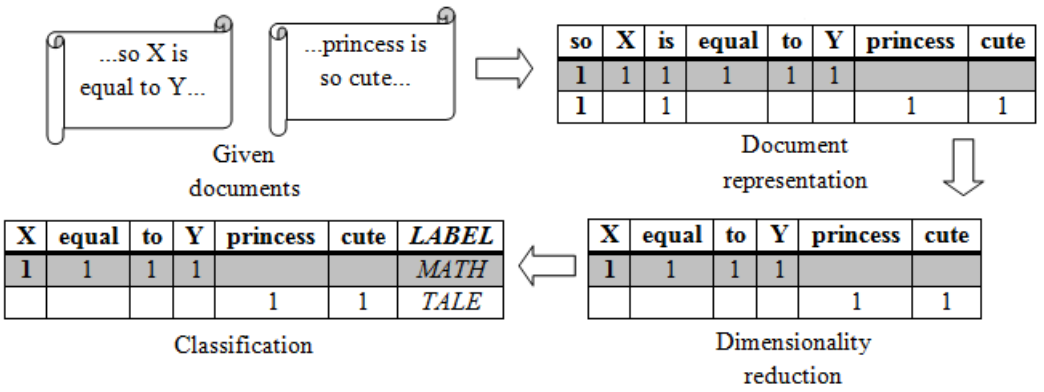

Figure 1.4: Example of text classification

some dimensionality reduction algorithm has been applied, in this case the words that appear in both documents simultaneously are considered irrelevant to the classification task (the words: "so" and "is") and have been removed from the feature space. Finally, the classification algorithm has been applied to the vectors: "111100" and "000011".

State-of-the-art approaches to text preprocessing usually involve the choice of the feature space. Firstly, words are transformed into the basic form using linguistic analysis. Secondly, the words that are irrelevant with respect to the given classification task are removed from the term set using expert knowledge and/or stop word lists that are specific for the given domain. There are also several ways to deal with synonymy and polysynonymy, some of them use existing dictionaries, which means that this approach is not transportable to another language with no available synonymic dictionary. Then, the remaining terms are weighted according to their "importance" to the domain and task. It can be done without the use of the class labels in train set (unsupervised term weighting) or with this information (supervised term weighting). The unsupervised approach to text preprocessing such as TF-IDF (term frequency inverse document frequency) is less effective than the supervised methods since the available information is not used. The problem of the supervised term weighting methods such as ConfWeight proposed by Soucy & Mineau (2005) is related to the required computational time due to the time-consuming statistical calculations such as Student distribution calculation and confidence interval definition for each term. The output of the text preprocessing algorithm is in most cases a collection of real-valued vectors in the chosen feature space.

The first aim of this thesis is the exploration of novel methods of text preprocessing: term weighting and feature space reduction. It should be mentioned that if linguistic analysis is applied to the given data, the performance may significantly increase. However, such analysis requires a lot of expert time and linguistic knowledge since it is specific. Furthermore, it is poorly transportable to another task nor another language. This thesis is concerned with the

statistical preprocessing techniques, which automatically extract information about relations between terms and categories and without being depended on a-priori task-specific knowledge. It is a non-trivial task to compete with algorithms which have been built on the knowledge of human experts (for example: algorithms that reduce feature space using stop word list).

In this work a novel supervised method of term weighting for text preprocessing is proposed. The idea of the method is based on an analogy between the term relevance value and a fuzzy rules membership function.

Many of the algorithms proposed in this thesis include the idea of combining supervised and unsupervised learning models to achieve better performance or to reduce the feature space dimensionality.

Two approaches for dimensionality reduction of the feature space are proposed and implemented in this work. These methods are based on feature selection and feature extraction using different term weighting techniques. The idea of the feature extraction algorithm is to combine synonym or nearly synonym words in one set on the basis of the statistics of their occurrence in the training data. First, all extracted terms are separated according to their relative frequencies to the category they most likely "belong" to. Then the clustering algorithm is applied to the terms of different categories separately, common weights of the term clusters are counted and using the obtained term clusters weights the new document representation is calculated. The feature selection method uses term filtering according to the chosen term weights. First, for each extracted term a real-valued weight is calculated using one of the term weighting formulas. Second, depending on the features of the text collection a set of borderline values is found, where the borderline value means that if the term weight is greater than the borderline, than the reduced feature space will include this term, otherwise the term will be excluded.

After clustering of terms their common weights might not be optimal. In order to improve the performance the common weights of the obtained term clusters are optimized using heuristic algorithm since it is not clear how the optimization criterion depends on the chosen weights. As genetic algorithms are well known for their applicability in the optimization tasks with not analytically defined criterion, they can be useful to optimize document representations. In this thesis the cooperative scheme of the coevolutionary genetic algorithm is applied since the formulation of the optimization task is naturally separable into the set of subtasks which can be optimized simultaneously. To provide the local convergence of the genetic algorithm a hybridized modification is used. The local search is performed only on the best obtained candidate solution for each generation.

The second aim of this thesis lies in the field of text classification and involves two main problems: clustering of non-uniform classes, which influence the document representation and improves performance of classification algorithms, and optimization of parameters of the SVM

and ANN-based classifiers using heuristic methods.

The idea of the classification improvement technique introduced in this thesis is based on the assumption that for classification algorithm it is easier to detect a small set of documents with high density rather than a large set which consists of many small subclasses (non-uniform class). A hierarchical agglomerative clustering algorithm has been proposed for such task since the output of hierarchical algorithms is a dendrogram and algorithm performance using different numbers of subclasses can be explored without the need to repeat the calculations.

Another task is to improve the classification quality and minimize the human supervision in tuning parameters of the Support Vector Machine (SVM) and Artificial Neural Network (ANN). It is not clear how to choose parameters of the classifier if the user does not have experience in the field. As classifiers SVM and ANN have been selected due to their good performance on various classification tasks. The problem of choosing the kernel function and other parameters of SVM and choosing the structure and weight coefficients of ANN remains a non-trivial problem. Thus, it is useful to consider heuristic methods to generate these text classifiers. In this case the standard genetic algorithm has not been selected for classifiers optimization, because it is initially designed for binary optimization. On the other hand biology related algorithms such as Particle Swarm Optimization can deal with the real-valued variables, and thus they seem to be a better choice. Machine learning methods that are based on the cooperation of different optimization algorithms generally outperform each individual method. Therefore, in this thesis the metaheuristic optimization algorithm has been applied instead of random search or default parameters which presumably will perform fine. In this work the Co-Operation of Biology Related Algorithms proposed by Akhmedova & Semenkin (2013) is modified to generate and optimize support vector machines and artificial neural networks. It is based on cooperation of biology inspired algorithms such as Particle Swarm Optimization [Kennedy *et al.* (1995)], Wolf Pack Search Algorithm [Yang *et al.* (2007)], Firefly Algorithm [Yang (2009)], Cuckoo Search Algorithm [Yang & Deb (2009)] and Bat Algorithm [Yang (2010)].

1.2 Main Contributions

The main contributions of this thesis fall into two fields:

1. Text preprocessing

   (a) Term weighting
   A novel method of supervised term weighting is proposed, which is based on statistics of term occurrence in each of the categories and adopts a modified formula for calculation of the fuzzy rules membership function. The introduced approach

performs better than standard techniques (such a various number of TF-IDF modifications) and requires significantly less computational time than the popular supervised term weighting algorithm called ConfWeight (3-4 times less depending on the given task).

(b) Dimensionality reduction based on term clustering
In this work a novel method that combines supervised and unsupervised techniques in order to reduce the feature space dimensionality is introduced. The idea of the proposed approach is to split the extracted term clusters in several groups according to the category with the greatest weight and to apply a cooperative scheme of the evolutionary genetic algorithm to find optimal weight for each term cluster. The method consists of two main stages. First one is a clustering of the terms for each category separately. Second stage is an optimization of the common weights of the obtained term clusters. This method reduces the term set (feature space) from about 250000 terms to 200 terms without a significant loss of the classification quality. In this thesis this dimensionality reduction approach has been compared with another method based on feature selection, where the reduced term set is obtained using the term filtering algorithm with different filter levels.

(c) Clustering non-uniform classes
A novel semi-supervised approach to improve the classification quality is introduced. It is designed to improve the detection of the large class that has items from different sources. Since in the initial problem statement all these items belong to the one class, it becomes difficult for the majority of the classification algorithms to detect this large non-uniform class. In this thesis several unsupervised algorithms including genetic algorithm with integers, learning vector quantization algorithm optimized with genetic algorithm and hierarchical agglomerative clustering method with different linkage criteria such as single-linkage, complete-linkage, average-linkage and Ward's criterion are applied to divide this large class into the set of subclasses in order to improve classification quality. The biggest improvement obtained on the SpeechCycle corpus is achieved using the hierarchical agglomerative clustering with the Hamming metric and the Ward criterion.

2. Text classification

(a) Optimization of the SVM and ANN-based classifiers for text classification
A novel approach to generate and optimize an Artificial Neural Network (ANN) and Support Vector Machine (SVM) classifiers is described. It is a metaheuristic method called the Co-Operation of Biology Related Algorithms (COBRA) that

uses a cooperation of five existing heuristic algorithms Particle Swarm Optimization, Wolf Pack Search Algorithm, Firefly Algorithm, Cuckoo Search Algorithm and Bat Algorithm. In this work three modifications of COBRA are used: original COBRA (unconstrained real-valued optimization), binary COBRA (unconstrained binary-valued optimization) and constrained COBRA (constrained real-valued optimization). The obtained ANN and SVM based classifiers perform better on various text classification tasks with different text preprocessings than the standard variants of ANN and SVM available in RapidMiner and other state-of-the-art classification algorithms.

1.3 Organization of the Thesis

This thesis consists of six chapters.

Chapter 1 states the main objectives and motivation for the proposed text preprocessing and classification algorithms. Chapter 2 describes the background and the state of the art on text preprocessing methods including algorithms for feature extraction, term weighting and dimensionality reduction, also discusses supervised and unsupervised classification methods and, finally, provides a brief description of heuristic optimization methods applied in this thesis to generate text classifiers.

In Chapter 3 text classification tasks and corpora used for training and evaluation are presented in detail. Differences between all corpora and important features are discussed.

Chapter 4 describes the main contributions of the thesis to the field of text preprocessing. First, a novel supervised term weighting method based on the adapted formula of the fuzzy rules relevance calculation is introduced. Several simple decision rules which can be applied for classification are discussed. Then, a novel approach to reduce feature space dimensionality is described. It is a feature extraction algorithm based on hierarchical term clustering according to categories and weights of terms, which is used to identify clusters of synonyms. In the same chapter a combination of unsupervised and supervised learning models to reduce feature space dimensionality and to improve classification results based on prior term clustering and weights optimization of the reduced term set is proposed. Finally, a semi-supervised modification of optimization criterion is introduced. In this chapter a dimensionality reduction method based on term filtering is described. This feature selection approach, where a set of borderline values is empirically selected for each text preprocessing techniques and for each database, is discussed in detail. Chapter 4 also describes contributions to the text classification field. First, an idea of classification improvement by applying clustering algorithm to the category with documents from different sources is introduced. Second, an application to text classification of metaheuristic approach for Support Vector Machines and Artificial

Neural Networks generation using the coevolutionary optimization method called COBRA (Co-Operation of Biology Related Algorithms) is presented in detail.

In Chapter 5, implementation details of applied methods are described and simulation results of the proposed text preprocessing algorithms in comparison with the state of the art methods are provided and discussed.

Finally, Chapter 6 provides a summary of the thesis contributions, describes the main conclusions derived from the thesis work and outlines open issues for future work.

Chapter 2

Background and State of the Art

This chapter provides an overview of the most common unsupervised and supervised text preprocessing methods. It describes some ways to reduce the dimensionality of the feature space with the focus on term clustering, since in this thesis clustering algorithms are applied to the term set. Most popular classification algorithms and their application to text categorization are discussed in detail, as well as performance measures, which are usually used to evaluate the algorithm results. Then several unsupervised techniques are described and some ways of combining them with supervised algorithms. Finally, a brief description of heuristic algorithms used in this thesis to train text classifiers is provided.

Nowadays, Internet and the World Wide Web generate a huge amount of textual information. It is increasingly important to develop methods of text processing such as text categorization. Text categorization can be considered to be a part of natural language understanding. Based on a set of predefined categories the task is to automatically assign new documents to one of these categories. There are many approaches to the analysis and categorization of text, and they could be roughly divided into statistical approaches, rule-based approaches and their combinations. Furthermore, the method of text preprocessing and text representation influences the results that are obtained even with the same classification methods.

Many researchers have used a number of unsupervised techniques for various tasks in order to improve classification quality. One common approach is to induce word features with unsupervised methods (for example, clustering which was used by Liang (2005); Koo *et al.* (2008); Ratinov & Roth (2009); Huang & Yates (2009) or neural language models, which have been proposed by Schwenk & Gauvain (2002); Mnih & Hinton (2007); Collobert & Weston (2008)) and then apply supervised algorithm.

Turian *et al.* (2010) have suggested learning unsupervised word features in advance without the task or model related information and combine and integrate them into existing Natural Language Processing systems. Despite an obvious advantage of this approach – word features

can be directly taken and used by many researchers – the performance might not be as good as the one obtained by a semi-supervised algorithm that learns unsupervised word features using task-specific information as in the semi-supervised models such as in Ando & Zhang (2005), Suzuki & Isozaki (2008), and Suzuki *et al.* (2009).

A number of works have recently been published on natural language call classification. Chu-Carroll & Carpenter (1999), Lee *et al.* (2000) and Gorin *et al.* (1997) proposed approaches using a vector-based information retrieval technique, the algorithms designed in Wright *et al.* (1997) use a probabilistic model with salient phrases. Schapire & Singer (2000) focused on a boosting-based system for text categorization.

In the field of natural language call routing the work most similar to this study has been done by Evanini *et al.* (2007), Suendermann *et al.* (2009) and Albalate *et al.* (2010). These papers report on text classification algorithms applied to the data from the same source - interactions with a troubleshooting agent. The focus of their work is on semi-supervised text classification methods applied to the corpus with only few labelled utterances for each problem category. The idea of the introduced algorithms is a document categorization into a set of clusters and using the available class annotations to match the obtained clusters and the given classes. Several text classification methods that can be applied to large scale data have been compared.

The information retrieval approach for call routing is based on a training of the routing matrix, which is formed by the statistics of appearances of words and phrases in a training set (usually after morphological and stop-word filtering). The new caller request is represented as a feature vector and is routed to the most similar destination vector. The most commonly used similarity criterion is the cosine similarity. The performance of systems that are based on this approach often depends on the quality of the destination vectors.

Text mining differs from the general classification in the way that the given textual documents have to be preprocessed first. This preprocessing stage converts the textual data into a format where the most significant features which help to identify the document from one category from another one are captured. The topic categorization can be considered as a preprocessing stage and for further classification. The preprocessing techniques can be divided into linguistic, statistical and combination of both. The linguistic preprocessing algorithms require the knowledge of semantic rules and language lexicon, which are usually specific to a particular language and language style. Such linguistic analysis is a very time consuming task and usually cannot be reused for another language, different style or domain. Statistical methods of text preprocessing are generally language and domain independent and, therefore, can be applied to cross-lingual, multi-lingual or unknown lingual textual corpora. This work is concerned only with the statistical approach to text classification.

2.1 State of the Art in Text Preprocessing Methods

2.1.1 Introduction

In order to solve any text classification problem a rather important step is to preprocess textual data into a form that classification algorithms are able to understand. It is well known that a way to represent a document influences the performance of text classification algorithms. The quality of the given text preprocessing can be measured according to the performance of classification algorithms that use this preprocessing. However, the preprocessing techniques and classification algorithms influence each other, some classifiers work better with one kind of preprocessing, other ones - with a different preprocessing type. Therefore, in order to evaluate the text preprocessing technique different text classifiers should be combined with this document representation.

In general, documents are not represented as sequences of symbols. They usually are transformed into vector representation in $\mathbb{R}^n$ (also called the feature space) because most of machine learning algorithms are designed for the vector space model. Such a document mapping into the feature space remains a complex task. Many researchers develop novel algorithms for text preprocessing, which are often specific for a given text classification problem. The simplest text preprocessing techniques are a bag-of-words model that represents the text data as a vector of numbers of times the term occurs in the document and a binary representation where each document is represented as a binary vector. In the binary preprocessing the i^{th} vector's component is equal to 1 if the i^{th} term occurs in the text document and equals to 0 if it does not. The most common text preprocessing method is Term Frequency Inverse Document Frequency (TF-IDF). Since it is borrowed from the information retrieval field, TF-IDF is initially designed for unsupervised learning. However, in supervised text classification information about class annotations is given and, therefore, the use of such information can be beneficial. In Z.-H. Deng & Xie (2004), Debole & Sebastiani (2003), Z.-H. Deng & Xie (2004), Batal & Hauskrecht (2009) and Soucy & Mineau (2005) unsupervised and supervised text preprocessing methods are tested and evaluated on different benchmark corpora.

Text preprocessing consists of feature extraction, term weighting and dimensionality reduction. In the feature extraction step the decision as to what will be considered as features is made. Features can be words, phrases, sets of synonyms etc. At this stage, words are usually transformed into the basic form and some words that have no impact on the classification process are removed. The term weighting means evaluating the "importance" of the extracted features. It can be done using statistical information according to the chosen weighting function. Finally, in the dimensionality reduction step the subset of the feature space is chosen to reduce the dimensionality. The main stages of text preprocessing are presented in Figure 2.1.

In this section the most common text preprocessing techniques are described in detail with

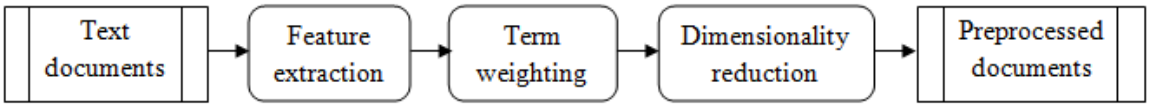

Figure 2.1: Main stages of text preprocessing

a focus on term weighting and dimensionality extraction.

2.1.2 Applications of Text Preprocessing and Text Classification

Text preprocessing and text classification algorithms play an important role in many commercial and scientific applications.

- **Biomedical Data**
  Text processing techniques enable biomedical researchers to effectively and efficiently access the knowledge hidden in large amounts of literature and contribute to classification of other biomedical data such as genome sequences, gene expression data and protein structures to support and accelerate biomedical discovery. In this area a lot of research has been done in modifying the standard text processing methods in order that they can perform in various tasks from the biomedical domain such as recognition of various biomedical entities and their relations, text summarization and question answering.

- **Social Media**
  Social media is one of the most common text sources in the Internet, where users are able to easily express themselves in the context of many subjects. Nowadays commercial sites use social media to influence potential customers and for targeted marketing. Processing text in social media creates a need to work with dynamic data that often uses poor and non-standard vocabulary. Moreover, since this data often occurs in the context of linked social networks, text preprocessing and text classification algorithms can use these links to improve the performance. Methods that consider not only content of the document but also such links provide more effective results than the methods that use only content or only link.

- **Multimedia Networks**
  Another common source of the text is the context of multimedia sharing sites such as Flickr or Youtube. In this case text processing and text classification methods have to take advantage from other domains and use this information together with the text collection. This problem is clearly related with the transfer learning field that focuses on including the data from other databases to the target domain.

- **News filtering and Organization**

A large amount of text information occurs in the context of the digital news services. Since the number of news articles increases dramatically every day, it becomes complicated to manually manage this document collection. Text preprocessing and text classification methods that can automatically process news articles, detect their topics and classify them according to their semantics are very useful applications.

- **Document Organization and Retrieval**
  A need to automatically organize large document collections occur not only in the field of news articles, but in many other domains including large digital libraries of documents, web collections, scientific literature etc. The most common approach is a hierarchical organization of document collections, because it is particularly useful for browsing and document retrieval.

- **Opinion Mining**
  Nowadays a lot of companies are interested in collecting and summarizing the clients opinions and reviews about their products. The field of opinion mining (also called sentiment analysis) focuses on the text processing algorithms that specialize on such opinionated data.

- **Email Classification and Spam Filtering**
  The organization of email is also an important application related to the document organization field. The most common tasks are to detect spam letters and to classify email according to the subject, topic or semantics.

There are clearly other applications of text preprocessing and text classification. Since these fields are growing rapidly and many text-related tasks become difficult to perform manually, a lot of research is focused on creating and adapting algorithms that can be useful in these areas.

2.1.3 Feature Extraction

Text preprocessing starts with the choice of feature space (vector-space model), because classification algorithms cannot handle text in a raw form. Text preprocessing methods are based on the assumption that the class of the document depends on the words contained in this document and, thus, the feature space should depend on the words occurred in the document collection.

The main aim of data preprocessing is to reduce the dimensionality of the feature space by controlling the size of the system's vocabulary (the number of terms). In some situations data preprocessing does not only reduce the complexity of the task, but also improves the performance, since the preprocessing can make the data more uniform.

In order to reduce the dimension and to increase the informativity different techniques are usually applied even before the text convertion into a vector space model.

Firstly, the so-called "stop-words" and "ignore-words" lists are collected. They usually contain prepositions, conjunctions, pronouns etc., those are the most frequent words without semantic importance. There are also domain-related "ignore-words", which are irrelevant according to the context of the categorization task. The design of these lists requires extensive expert knowledge and domain-related lists are usually not transportable to another task. Nowadays, researchers tend to focus on text preprocessing techniques which are able to automatically detect "stop-words" and "ignore-words" and to assign them low enough weights in order to avoid the machine learning algorithms to be affected by them.

Secondly, all terms are normalized which means that words are converted into a basic form where suffixes and prefixes are removed (it is also called stemming) and sometimes all synonyms are replaced by one single word.

Numbers and non-letter characters are often removed or substituted for a special token. Case and special characters are unified. This type of preprocessing is needed in some languages which contain special characters (for example German has umlauts 'ä', 'ö', 'ü' and special character 'ß' which can be written as 'ae', 'oe', 'ue' and 'ss' accordingly) since these special characters are used in some documents and are replaced with the equivalents in other documents.

It should be mentioned that in some cases a feature is not one word but a collection of all synonyms or fixed phrases and proper nouns. This collection is considered as an inseparable unit and is also called a term. On the level of machine learning algorithms it is not important what was considered as a term. Machine learning algorithms process terms as features.

Different approaches to text preprocessing can be classified by

1. different ways to understand what a term is;

2. different ways to compute term weights.

Words, phrases or clusters of synonyms are often considered as features or terms. Usually one identifies the term set as all words that appeared in the training data. This model is called a bag of words or a set of words representation, depending on whether weights are binary or not. Key phrases are also used to index the document or as features in further processing (e.g., Ong & Chen (1999), Turney (2000) and Li *et al.* (2010)).

Before indexing, the removal of function words (i.e., topic-neutral words such as articles, prepositions, conjunctions, etc.) is almost always performed (exceptions include Lewis *et al.* (1996), Nigam *et al.* (2000) and Riloff (1995)). Concerning stemming (i.e., grouping words that share the same morphological root), its suitability to text classification is controversial. Although, similarly to unsupervised term clustering of which it is an instance, stemming has

sometimes been reported to hurt effectiveness (e.g., Baker & McCallum (1998)), the recent tendency is to nevertheless adopt it, as it reduces both the dimensionality of the term space and the stochastic dependence between terms.

2.1.3.1 Stop Word Filtering

Stop word filtering is based on the manually created lists of non-informative words (for example, the SMART stop list Blanchard (2007); Dragut *et al.* (2009). In this respect, it differs from the feature selection methods that attempt to remove non-informative terms automatically.

Stop words are irrelevant for information extraction, clustering or classification, since they do not contain any semantic information related to the document category. Pronouns and prepositions are examples of such stop words. For natural language documents (usually documents which have been transcribed from speech automatically by an ASR) the sounds which are common for a spontaneous speech ("eh", "ehm", "uh" etc.) also considered stop words.

These lists are obviously different for every language and domain and creating such list for a new domain requires a lot of time and knowledge.

2.1.3.2 Stemming

The goal of stemming is to transform different forms of the word (inflected or derived) to its "canonical" form: stem or lemma. It eliminates the morphological diversity which is provided by the suffixes and prefixes as discussed in Porter (1980); Schmid (1994).

There are several approaches to stemming for example lookup algorithms, suffix-stripping algorithms, lemmatization algorithms, n-gram analysis and a hybrid approach.

There are two error measures in stemming algorithms: overstemming and understemming. In the case of overstemming, the morphologically related words, which are used in different context with different meaning are considered to be the same stem. In this situation the relevance of the obtained feature set will be worse. Understemming is the case where two words should be stemmed to the same base but the stemming algorithm has not stemmed them to one root.

2.1.4 Term Weighting

Term weighting is a part of text preprocessing, where the "importance" values for the extracted features are measured. There are different ways to calculate the weight of each word. The term weighting methods can be roughly divided into two groups:

- unsupervised term weighting
  These methods do not use the information about class labels of the document. The

weights are calculated on the basis of only statistics on the term occurrence in documents from the whole collection.

- supervised term weighting
  The term weights are calculated using the statistical information about the class labels.

In general, document representation consists of two parts: the one that corresponds to statistics of term occurrences in the given document (binary, term frequency, random walk etc.) and the part that estimates statistics of term occurrences in the whole collection (IDF modifications).

2.1.4.1 Unsupervised Term Weighting

The standard term weighting methods belong to the unsupervised term weighting since they have been initially designed for Information Retrieval.

The simplest way to measure the "importance" of terms is the bag-of-words model or binary feature vectors where the weight of each term equals to 1.

Bag-of-Words Model and Binary Representation

An early reference to "bag of words" in a linguistic context can be found in Harris (1954). In the bag-of-words model text documents are represented as feature vectors where each term is assigned to a vector component and a value of this component is a number of times this term occurs in the text document.

For text classification simpler binary feature values (i.e., a word either occurs or does not occur in a document) are often used instead.

The simplest text representation is a binary vector where each word that occurred in the training data is taken as a coordinate (1 denoting presence and 0 denoting absence of the term in the document). The dimension of the feature space is the number of words appeared in the training set. According to this preprocessing each document d is represented by a binary vector $d = (x_1, x_2, \ldots, x_N)$, where the component $x_i = 1$ if the document d contains the i word and $x_i = 0$ if the document d does not include this word.

Since each term weight in the binary representation is equal to one and term frequencies occurred in the document are not considered, there are several problems with this approach:

- Feature space dimensionality is large and since all term weights equal 1, there is no automatic way to distinguish terms from each other and, therefore, to reduce the term set. Some words have no meaning according to the given classification task, however, in binary representation they are still considered as features. There are many synonyms in the feature space and instead of replacing all synonyms with one term they are separate features.

- The feature space is sparse. There are usually many zero coordinates in a particular document since all words that appeared in the whole collection of documents are considered as features.
- Since all terms appeared in the given document have the same impact, it makes it difficult for classification algorithms to process such vectors. It is particularly complicated in case of long documents with many unequal terms. However, in case of short documents with only 5-6 terms (only key words) binary representation can produce good enough performance.

There are other text preprocessing techniques that attempt to overcome those problems.

Term Frequency

In this preprocessing scheme, which is the most common in text classification, the term weight in a document increases with the number of times that the term occurs in the document and decreases with the number of times that the term occurs in the corpus. This means that the importance of a term in a document is proportional to the number of times that the term appears in the document and while the importance of the term is inversely proportional to the number of times that the term appears in the entire corpus.

The standard variant of the term frequency is calculated according to the following equation.

$$\mathrm{TF}_{ij} = \frac{t_i}{\sum_{l=1}^{T} t_l}, \tag{2.1}$$

where t_i is the number of times the ith word occurs in the jth document and T is the document size.

The normalized version of the term frequency is shown in the following equation

$$\mathrm{T}F_{norm} = 0.5 + 0.5 * \frac{\mathrm{t}f_{ij}}{\mathrm{atf}}, \tag{2.2}$$

where atf is an average frequency of the term in the document.

There are several modifications of TF which are often used in the literature. Term frequency can be also calculated using Equations 2.3, 2.4 and 2.5.

$$\mathrm{TF} = 1 + \log(\mathrm{t}f_{ij}), \tag{2.3}$$

$$\mathrm{TF} = (1 + \log(\mathrm{t}f_{ij}))/(1 + \log(\mathrm{mtf})), \tag{2.4}$$

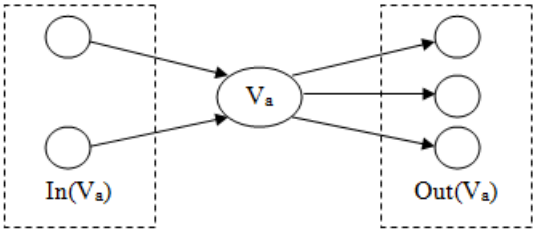

Figure 2.2: Model of the graph $G = (V, E)$

$$\mathrm{TF} = \log(\mathrm{t}f_{ij} + 1), \tag{2.5}$$

where $\mathrm{t}f_{ij}$ is calculated as in 2.1 and mtf is a maximal term frequency in the document.

The term frequency itself seems to be very intuitive, however one can notice a major drawback. For example, if a term occurs often in the given document, then it will have a high frequency, but since this word can occur in nearly all documents, it carries no information about the semantics of a document (this situation often happens with nouns, prepositions etc.). These circumstances correspond to the well known Zip-Mandelbrot law [Mandelbrot (1961)], which states, that the frequency of terms in texts is extremely uneven. Some terms occur very often, whereas as a rule of thumb, half of the terms occur only once.

Random Walk

This approach uses term co-occurrence as a measure of the dependency between word features. In this model terms are considered to be a graph, where vertices represent terms and edges are relations between terms. This idea has been applied for text classification in Hassan *et al.* (2007).

Let $G = (V, E)$ be a graph, where V is a set of all vertices and E is a set of all edges; $\mathrm{In}(V_a)$ is a subset of all vertices that point to the vertex V_a (predecessors), and $\mathrm{Out}(V_a)$ is a subset of all vertices that the vertex V_a points to (successors) (Figure 2.2).

The value $S(V_a)$ associated with the vertex V_a is calculated using a following recursive function:

$$S(V_a) = (1 - d) + d \cdot \sum_{V_b \in \mathrm{In}(V_a)} \frac{S(V_b)}{|\mathrm{Out}(V_b)|}, \tag{2.6}$$

where d is a parameter which can be set between 0 and 1 (usually $d = 0.85$).

The numerical results reported in Hassan *et al.* (2007) have demonstrated the classification improvement over traditional term frequency with the NaiveBayes, SVM, k-NN and Rocchio classifiers.

Term Frequency Inverse Document Frequency (TF-IDF)

All unsupervised text preprocessing methods are calculated as a multiplication of term frequency $\mathrm{t}f_{ij}$ (ratio between the number of times this word occurs in the document and the document size) and inverse document frequency $\mathrm{id}f_i$.

The most commonly used approach is TF-IDF. There are many variants of the TF-IDF weighting schemes. In the following several popular variants are presented.

1. Firstly, the RSJ weight (Robertson and Sparck Jones) that has been proposed in Robertson & Jones (1976) should be mentioned. It is calculated according to the following equation

$$\mathrm{id}f_i = \log \frac{|D|}{n_i}, \tag{2.7}$$

where $|D|$ is the number of document in the training data and n_i is the number of documents which have the ith word (n_i can be also calculated as the number of times ith word appears in all documents from the training data).

2. One of the popular versions is known as best-match weighting function BM25 (see Robertson (2004); Robertson *et al.* (1995)) and it is calculated using the following equation

$$\mathrm{id}f_i = \frac{(k_1+1)\cdot \mathrm{t}f_i}{K+\mathrm{t}f_i} \log \frac{|D|}{n_i}, \tag{2.8}$$

where $k_1$and K are global parameters that are generally unknown but may be tuned on the basis of evaluation data.

3. Another idea is to introduce the real-valued parameter α instead of the logarithm. It is given by

$$\mathrm{id}f_i = \left(\frac{|D|}{n_i}\right)^{\alpha}, \alpha \in (0,1), \tag{2.9}$$

where n_i is the number of documents which have the ith word (n_i can be also calculated as the number of times ith word appears in all documents from the training data) and α is the parameter (in this thesis $\alpha = 0.1, 0.5, 0.9$ are used).

4. Term Frequency Inverse Corpus Frequency (TF-ICF) which has been propposed in Reed *et al.* (2006). This term weighting algorithm uses $\log(1+\mathrm{t}f_i)$as a term frequency part and $\log\left(\frac{N+1}{n_i+1}\right)$ as a weight of terms. It is given by

$$\mathrm{tf.ic}f_i = \log(1+tf_i)\cdot \log\left(\frac{N+1}{n_i+1}\right), \tag{2.10}$$

where N is the number of documents in the corpus and n_i is the number of documents which contain the term i.

5. Term Weight Inverse Document Frequency (TW-IDF) is another unsupervised term weighting method which has been proposed in Rousseau & Vazirgiannis (2013) with the graph-of-word model that captures the relationships between the terms using an unweighted directed graph of terms. This term weighting scheme is calculated as follows:

$$\text{TW-IDF}(t,d) = \frac{\text{tw}(t,d)}{1-b+b\times\frac{|d|}{\text{avdl}}} \times \log\frac{N+1}{\text{df}(t)}, \tag{2.11}$$

where $\text{tw}(t,d)$ is the weight of the vertex t (term t) in the graph-of-word representation of the document d, b is a parameter ($b = 0.003$), N is the number of all documents, $\text{df}(t)$ is the document frequency of the term t across all documents, avdl is the average document length and $|d|$ is the length of the document d.
The experiments conducted in Rousseau & Vazirgiannis (2013) have shown that this approach outperforms BM25 on different benchmark corpora.

6. Zhang & Nguyen (2005) have presented a new measure of weighting term significance which can be calculated using the following equation

$$w_{ik} = c^{-(f_{ik}-f_{ia})^2} \cdot \log\left[\frac{N\cdot D_k}{d_k\cdot L_k}\right], \tag{2.12}$$

where f_{ia} is the average value of the frequency range in document i; f_{ik} is the raw frequency of the term k in document i; L_k is the number of times term k appeared in the whole corpus; D_k is the number of all terms in documents which contain the term k; w_{ik} is the term significance of term k in document i; $c > 0$ is a constant which influences the impact of term frequencies on the weight; N is the number of documents in corpus; d_k is the number of documents containing term k.
The numerical experiments conducted in Zhang & Nguyen (2005) have shown that the introduced term weighting has achieved better performance than the Salton's algorithm and the same performance as Sparck Jones' method.

7. Another method called Modified Inverse Document Frequency (MIDF) has been proposed in Deisy *et al.* (2010) and is defined as follows.

$$\text{MIDF}(i,j) = [1+\log 2\{\text{TF}(i,j)\}]\cdot\frac{\text{DFR}(i)}{\text{DF}(i)}, \tag{2.13}$$

where the document frequency $\text{DF}(i)$ is the sum of the term frequency and $\text{DFR}(i,j)$ is the number of non-zero values of the document i.
The performance of the introduced MIDF term weighting have been compared with

other techniques in combination with SVM classifier. The results have shown that the MIDF has achieved better performance.

There are many different unsupervised term weighting techniques which have not been mentioned before. One of them is a term weighting method proposed in Ropero *et al.* (2009) that takes advantage of the use of a fuzzy logic based term weight as an alternative to the standard TF-IDF. This approach gives better results, especially in terms of extracting the related information as well.

However, the TF-IDF representation remains the standard preprocessing scheme in text classification and information retrieval and most researchers use it to solve the text categorization tasks.

2.1.4.2 Supervised Term Weighting

Traditional term weighting schemes are usually borrowed from the Information Retrieval field. In contrast to the information retrieval that uses only statistics of the term occurrence in all documents of the given corpus, the text classification is a supervised learning that uses the prior information on the training documents membership to the pre-existing categories.

The supervised term weighting methods generally use the information about term-document co-occurrence in several ways.

1. Term weighting using the feature selection function (information gain, gain ratio, Odds Ratio, and others)
   The first approach to the supervised term weighting is to apply the feature selection scores. Feature selection is usually performed to reduce the dimensionality of the initial term set by the selection the most relevant and important features. The terms that have higher feature selection score are considered to influence more to the classification process than the ones with lower scores. These functions are very effective for term weighting task since they have been designed for the detection of the important terms. Z.-H. Deng & Xie (2004) have applied χ^2 score instead of IDF to weight terms and TF. The results using SVM classifier have shown that the χ^2 weight produces better performance than the IDF weight.
   Debole & Sebastiani (2003) have used information gain, χ^2 and gain ratio as term weights. Experiments conducted in that work have not shown the superiority of the supervised weights over the usual unsupervised IDF. Moreover, TF-IDF outperforms TF$\cdot\chi^2$ that contradicts the result obtained in Z.-H. Deng & Xie (2004).
   Batal & Hauskrecht (2009) have shown that by using information gain and χ^2 metric as a supervised term weight the performance of the k-nearest neighbor classifier (k-NN) can be significantly improved.

2. Term weighting based on statistical confidence intervals
   This approach uses the prior knowledge of the statistics in the labelled data. One of the well known term weighting based on this idea is ConfWeight proposed by Soucy & Mineau (2005).

3. Term weighting combined with the classifier
   The idea is similar to the first approach, but in this case the function which the text classifier uses to detect the classes is assumed to produce appropriate term weights.
   These methods weight terms using the selected text classification algorithm. Since the classification algorithm detects a category of the document by assigning different scores to the test documents, these scores can be used to calculate the weights of terms. In Han *et al.* (2001) term weights are calculated using the k-NN algorithm at each step iteratively. The weights are modified in each iteration and the classification accuracy is calculated on the development set. The weights converge to the optimum, however, since the term set size is usually very large, this method is too slow for real problems.

There are various supervised term weighting techniques which differ in the ways of the use of the information about the document membership. For example, for an opinion mining task a special variation of TF-IDF has been proposed in Ghag & Shah (2014). It is called SentiTFIDF and it classifies a term as negative or positive based on its proportional frequency and presence across positively labelled documents in comparison with negatively labelled documents.

In the following several important methods of the supervised term weighting are described in detail.

Confidence Weights (ConfWeight)

This approach has been proposed by Soucy & Mineau (2005). ConfWeight method has an important advantage: it implicitly does feature selection since all frequent words have zero weights. The main idea of the method is that the feature f has a non-zero weight in the class c only if the frequency of the feature f in documents of the c class is greater than the frequency of the feature f in all other classes.

First, the proportion of documents containing term t is defined as the Wilson proportion estimate $p\,(x, n)$ [Wilson (1927)] by the following equation

$$p(x,n) = \frac{x + 0.5z^2_{\alpha/2}}{n + z^2_{\alpha/2}}, \tag{2.14}$$

where x is the number of documents containing the term t in the given corpus, n is the number of documents in the corpus and $\Phi(z_{\alpha/2}) = \frac{\alpha}{2}$, where Φ is the t-distribution (Student's law) when $n < 30$ and the normal distribution when $n \geq 30$.

In this work $\alpha = 0.95$ and $0.5z^2_{\alpha/2} = 1.96$ (as recommended by the authors of ConfWeight). For each feature f and each class c two functions $p_+(x,n)$ and $p_-(x,n)$ are calculated. $p_+(x,n)$ is given by

$$p_+(x,n) = p(x,n), \tag{2.15}$$

where x is number of vectors which belong to the class c and have non-zero feature f; n is the number of documents which belong to the class c. $p_-(x,n)$ is given by

$$p_-(x,n) = p(x,n), \tag{2.16}$$

where x is number of vectors which have the feature f but do not belong to the class c; n is the number of documents which do not belong to the class c.

Confidence interval $(\underline{p}, \overline{p})$ at 95% is calculated using equations 2.17 and 2.18.

$$\underline{p} = p - 0.5z^2_{\alpha/2}\sqrt{\frac{p(1-p)}{n + z^2_{\alpha/2}}} \tag{2.17}$$

$$\overline{p} = p + 0.5z^2_{\alpha/2}\sqrt{\frac{p(1-p)}{n + z^2_{\alpha/2}}} \tag{2.18}$$

The strength of the term f in the category c is defined as equation 2.19.

$$\mathrm{str}_{f,c} = \begin{cases} \log_2(2\frac{\underline{p_+}}{\underline{p_+}+\overline{p_-}}), & \text{if } \left(\underline{p_+} > \overline{p_-}\right) \\ 0, & \text{otherwise} \end{cases} \tag{2.19}$$

Maximum strength of the term f is calculated as follows

$$\mathrm{Maxstr}(f) = (\max_{c \in C} \mathrm{str}_{f,c})^2. \tag{2.20}$$

ConfWeight method uses Maxstr as an analog of IDF and $\log(1 + \mathrm{t}f_i)$as a term frequency part. It is given by

$$\mathrm{ConfWeight}t_{ij} = \log(\mathrm{t}f_{ij} + 1) \cdot \mathrm{Maxstr}(i). \tag{2.21}$$

Maxstr (maximum strength) is a relatively uncommon but promising method for term weighting. It is an alternative to IDF. When calculating Maxstr the features are selected implicitly because frequent words are assigned the minimum weight. The formula of Maxstr is computationally harder and it is based on statistical confidence intervals. In ConfWeight method term weights are calculated using the statistical confidence and this calculation requires the prior knowledge of the class labels.

Numerical experiments conducted in Soucy & Mineau (2005) have shown that the Con-

fWeight method outperforms gain ratio and TF-IDF with SVM and k-NN as classification methods on three benchmark corpora. The main drawback of the ConfWeight method is computational complexity. This method is more computationally demanding than TF-IDF method because ConfWeight method requires time-consuming statistical calculations such as Student distribution calculation and confidence interval definition for each word.

Relevance Frequency (TF.RF)

Another supervised term weighting method has been proposed in Man Lan & Lu (2009). The idea of the method that the higher the difference between the concentration of the term with high frequency in the positive category and its concentration in the negative category, the bigger contribution it makes towards the detection of this category.

It can be calculated using the following equation

$$\text{tf.rf} = \text{tf} \cdot \log\left(2 + \frac{a}{\max(1, c)}\right), \tag{2.22}$$

where a is the number of documents in the positive category which contain this term and c is the number of documents in the negative category which contain this term.

The experimental results using linear SVM and k-NN algorithms which have been conducted in Man Lan & Lu (2009) have shown that TF.RF outperforms other term weighting schemes in all experiments.

Term Relevance Ratio (TRR)

Another term weighting scheme has been proposed by Ko (2012), which can be calculated using the following equation.

$$\text{t}w_i = (\log(\text{t}f_{ij}) + 1)) \cdot \log\left(\frac{P(w_i|c_j)}{P(w_i|\bar{c}_j)} + \alpha\right) \tag{2.23}$$

The term weighting part is calculated using the following formulas.

$$P(w_i|c_j) = \frac{\sum_{k=1}^{|T_{c_j}|} \text{t}f_{ik}}{\sum_{l=1}^{|V|}\sum_{k=1}^{|T_{c_j}|} \text{t}f_{lk}} \text{ or } P(w_i|c_j) = \sum_{k=1}^{|T_{c_j}|} P(w_i|d_k) \cdot P(d_k|c_j), \tag{2.24}$$

$$P(w_i|\bar{c_j}) = \frac{\sum_{k=1}^{|T_{\bar{c_j}}|} \text{t}f_{ik}}{\sum_{l=1}^{|V|}\sum_{k=1}^{|T_{\bar{c}_j}|} \text{t}f_{lk}} \text{ or } P(w_i|\bar{c_j}) = \sum_{k=1}^{|T_{\bar{c}_j}|} P(w_i|d_k) \cdot P(d_k|\bar{c_j}), \tag{2.25}$$

where $|V|$ is the number of terms extracted from the the training data; $|T_{c_j}|$ is the number of documents which belong to the category c_j and $|T_{\bar{c}_j}|$ is the number of documents which do not belong to the category c_j.

Experimental results show good performance of the introduced scheme in comparison with other term weighting techniques using k-NN and SVM on the two benchmark databases.

ConceptFreq

This term weighting method has been proposed in Song *et al.* (2004). It is based on the lexical chain and the sentence extraction algorithm that uses it. ConceptFreq is calculated as follows

$$\text{ChainFreq}(w_{ij}, c_j) = \text{WordConnectivity}(w_{ij}) \times \text{ChainConnectivity}(c_j), \tag{2.26}$$

where WordConnectivity (w_{ij}) is given by

$$\text{WordConnectivity}(w_{ij}) = \text{Frequency}(w_{ij}) \times \frac{r(w_{ij}, c_j) + \alpha}{\sum_{j=1}^{J} \sum_{k=1}^{N} r(w_{kj}, c_j)} \tag{2.27}$$

and ChainConnectivity (c_j) is given by

$$\text{ChainConnectivity}(c_j) = \frac{\sum_{k=1}^{N} r(w_{kj}, c_j)}{|c_j|}, \tag{2.28}$$

where $r(w_{ij}, c_j)$ is the number of relations that have w_{ij}in chain c_j, $|c_j|$ is the number of unique words in chain c_j and α is a constant.

Authors of Song *et al.* (2004) have shown that the proposed scheme outperformed the standard TF-IDF, however the problem of semantic relations between terms remains a complex task.

2.1.5 Dimensionality Reduction

Text collections which contain millions of unique terms are quite common. Thus, in order to more efficiently apply existing classification algorithms and to achieve better performance, dimensionality reduction of the feature space is widely used when applying machine learning methods to text categorization. Since most classification algorithms cannot handle such high dimensionality of text data, techniques which reduce the term set size are particularly useful for text classification for both efficiency and efficacy.

In the literature, the methods which select a subset of the existing features and the methods which transform existing features into a new reduced term set are distinguished. Both types of methods can rely on a supervised or unsupervised learning procedure (Blum & Langley (1997), Sebastiani (2002), Christopher *et al.* (2008), Fodor (2002), Berkhin (2006) and Slonim & Tishby (2001)):

1. Feature selection (term selection): T_s is a subset of T. These methods usually consider the selection of only the terms that occur in the highest number of documents, or the se-

lection of terms depending on the observation of information-theoretic functions such as the DIA association factor, χ^2, NGL coefficient, information gain, mutual information, odds ratio, relevancy score, GSS coefficient and others.

2. Feature transformation (term extraction): the terms in T_p are not of the same type as the terms in T (the terms in T could be words, but the terms in T_p may not be words at all), the terms in T_p are obtained by combinations or transformations of the original terms. These methods generate from the original a set of "synthetic" terms that maximize effectiveness based on:

   - term clustering,
   - latent semantic analysis,
   - latent Dirichlet allocation,
   - principal component analysis and others.

   After such transformation it may be necessary to reduce the number of the new features using a selection method: a new term set T_{sp} that is a subset of T_p.

Dimensionality reduction is also beneficial since it tends to reduce overfitting, that is, the phenomenon by which a classifier is tuned also to the contingent characteristics of the training data rather than just the constitutive characteristics of the categories. Classifiers that overfit the training data are good at reclassifying the data they have been trained on, but much worse at classifying previously unseen data.

2.1.5.1 Feature Selection

Feature selection approaches attempt to find such subset of the terms initially extracted from the train sample that will lead to the better document representation and higher performance of the classification algorithm used to solve the task.

Yang & Pedersen (1997) have shown that term selection can increase the algorithms performance up to 5% depending on how many terms have been excluded, the selection criterion that has been applied and the classifier.

Moulinier & Ganascia (1996) have proposed a method of term selection which tunes to the classifier that is used. Algorithm 2.1 shows the scheme of the feature selection method.

An obvious advantage of this approach is that the document representation is tuned with the classifier and in the reduced term set there may be different numbers of terms for detecting each category (some categories are more separable than others). The problem is that the method is time consuming because the term set size is usually very large.

Feature selection algorithms can be divided by

Algorithm 2.1 Feature selection method

1. Start with the initial term set $T_r = T$.
2. Choose one term t.
3. Perform the classification using the term set T and reduced term set $T_r \setminus t$.
4. Choose the term set which produces better classification result.
5. Check the stopping criterion, otherwise go to the step 2.

- Filters
  These methods evaluate each term independently from the classifier according to the chosen weighting technique, rank the features after evaluation and take the subset with the highest weights.
- Wrappers
  In contrast to the previous approach these algorithms depend on the selected classifier, they evaluate subsets of the initial term set with the chosen classifier. The subset which provides the best performance is selected. Wrappers are more reliable than filters because classification algorithm influences the selection of the term subset. Since it is a time consuming process heuristic algorithms are often used to find the optimal subset for the selected classifier: genetic algorithm, greedy stepwise, best first or random search.
- Embedded Selectors
  Embedded algorithms attempt to select features during classification in a similar way as artificial neural networks.

The filtering approach John *et al.* (1994) is usually a computationally easier alternative than the embedded selectors, which remove terms with low measure of "importance". There are some ways to measure this "importance" of the term. The most natural approach for the term selection is a use of Inverse Document Frequency as a term "importance" measure.

Yang & Pedersen (1997) have shown that the filtering term reduction where the words which appear more often are kept in the reduced term set. Term set reduced 10 times will produce the same classification performance as the initial set, and 100 times reduction will lead only to a small loss of effectiveness.

This result seems to show that the words with the highest frequency are the most useful for text classification. However, there is somehow a contradictive result Salton & Buckley (1988) that shows that the words with lowest frequency carry the most information about document category.

Name	Author(s)	Formula
DIA association factor	Fuhr *et al.* (1991)	$P(c_i\|t_k)$
Information gain	Dumais *et al.* (1998) Joachims (1998) Yang & Pedersen (1997) Yang & Liu (1999)	$\sum_{c\in\{c_i,\bar{c}_i\}}\sum_{t\in\{t_k,\bar{t}_k\}} P(t,c)\cdot$ $\cdot\log\frac{P(t,c)}{P(t)\cdot P(c)}$
Mutual information	Yang & Pedersen (1997)	$\log\frac{P(t_k,c_i)}{P(t_k)\cdot P(c_i)}$
Chi-square	Yang & Pedersen (1997) Yang & Liu (1999)	$\frac{\lvert Tr\rvert\cdot[P(t_k,c_i)\cdot P(\bar{t}_k,\bar{c}_i)-P(t_k,\bar{c}_i)\cdot P(\bar{t}_k,c_i)]^2}{P(t_k)\cdot P(\bar{t}_k)\cdot P(c_i)\cdot P(\bar{c}_i)}$
NGL coefficient	Ng *et al.* (1997) Ruiz & Srinivasan (1999)	$\frac{\sqrt{\lvert Tr\rvert}\cdot[P(t_k,c_i)\cdot P(\bar{t}_k,\bar{c}_i)-P(t_k,\bar{c}_i)\cdot P(\bar{t}_k,c_i)]}{\sqrt{P(t_k)\cdot P(\bar{t}_k)\cdot P(c_i)\cdot P(\bar{c}_i)}}$
Relevancy score	Wiener *et al.* (1995)	$\log\frac{P(t_k\|c_i)+d}{P(t_k\|\bar{c}_i)+d}$
Odds ratio	Ruiz & Srinivasan (1999)	$\frac{P(t_k\|c_i)\cdot(1-P(t_k\|\bar{c}_i))}{(1-P(t_k\|c_i))\cdot P(t_k\|\bar{c}_i)}$
GSS coefficient	Galavotti *et al.* (2000)	$P(t_k,c_i)\cdot P(\bar{t}_k,\bar{c}_i)-$ $-P(t_k,\bar{c}_i)\cdot P(\bar{t}_k,c_i)$

Table 2.1: The most common term weighting formulas (main functions used for Feature Selection Sebastiani (2002))

There are other ways to weight the term importance in the corpus. Standard approach is to use an information-theoretic function, which has been used to weight the terms. Many information-theoretic functions have been used in the literature. A very detailed list is provided in Sebastiani (2002), which is repeated in Table 2.1.

These functions try to capture the intuition that the most important terms for the classification performance on the category c_i are the ones distributed most differently in the sets of positive and negative examples for the category c_i.

Karabulut *et al.* (2012) have shown how the chosen feature selection technique affects the classification performance of different algorithms: Naive Bayes, Multilayer Perceptron, and J48 decision tree are compared. The most sensitive classifier was multilayer perceptron.

Cannas *et al.* (2012) proposed a method to reduce feature space dimension based on the combination on filter and wrapper approaches. In this method the term relevance is weighted by a filter and, then, the top-ranked terms are grouped in several sets of relatively small size. The performance of the proposed approach has been evaluated on Reuters corpus and shows competitive results among existing models.

Figueiredo *et al.* (2011) generate new compound-features which are composed by terms that co-occur in documents. The idea is to reduce the ambiguity and noise in the bag of words model by using single-terms and compound features together. Experimental results

have shown a significant improvement with many classification algorithms.

Vieira *et al.* (2012) introduce an idea of fuzzy criterion in feature selection, which allows to define the goal in feature selection in a more flexible way and does not have a problem of weighting different goals. Ant colony optimization algorithm has been applied for the optimization problem.

2.1.5.2 Feature Transformation

Another strategy to reduce the dimensionality of the document space is feature transformation. The original terms may not have optimal dimensions for document content representation. Methods for term extraction try to solve these problems by creating artificial terms that do not suffer from them. These methods consist of

1. the method of constructing the new terms from existing ones;
2. the method of representing the documents based on new terms.

The most widely used methods of dimensionality reduction by feature extraction are term clustering and latent semantic indexing.

Term Clustering

Term clustering aims to combine terms with synonymous or near synonymous meaning into groups (clusters) and then this set of clusters (or centroids or a prototype) is used as a term set (feature space). Term clustering approach to feature space reduction differs from the term selection in the way that the term clustering groups terms which are related to each other on the basis of statistics of their appearance in the corpus, while term selection spots the non-informative terms and removes them.

The idea of term clustering was proposed by Brown *et al.* (1992) and applied in many different fields including speech recognition, named entity tagging, machine translation, query expansion, text categorization and word sense disambiguation. The idea of using class-based language model by applying term clustering, proposed by Momtazi & Klakow (2009), has been found to be effective in overcoming the problems of data sparsity, exact matching using small text segments.

Recently researchers have become interested in the area of term clustering and consequential reweighing of the terms. The new document clustering method has been proposed in Park *et al.* (2013). It uses the reweighed term based on semantic features for enhancing document clustering. It reduces the semantic gap between the user's requirement and the obtained clustering by using the document samples.

Lewis (1992) has first proposed the use of term clustering in text classification. The name of his method is reciprocal nearest neighbor clustering. Lewis has halved the term set

by combining pairs of the most similar terms (add the similarity measure). The algorithms performance on this reduced term set was worse than the results on the initial one probably due to the clustering method that was used.

In contrast with those unsupervised term clustering approaches, Baker & McCallum (1998) have proposed a supervised learning model applied to term clustering. The idea was that the terms which point to the same category or the group of categories should be combined in one cluster. As a classification algorithm they have used a Naive Bayes and its performance on the obtained reduced term set has shown some improvement. The work of Slonim & Tishby (2001) has also shown the advantages of the supervised term clustering.

In Section 2.3 unsupervised machine learning algorithms which can be applied to the term clustering are described.

Latent Semantic Indexing

Latent Semantic Indexing (LSI) proposed by Deerwester *et al.* (1990) or Latent Semantic Analysis (LSA) proposed by Landauer *et al.* (1998) are methods that attempt to overcome the problem of using the synonymous, polysemous words as dimensions of the feature space. This approach reduces the feature space by mapping the initial documents vectors of high dimensional space into the lower dimensional space where the new dimensions are obtained by combining initial terms according to their co-occurrence. According to Deerwester *et al.* (1990), this feature transformation can also help to remove the "noise" induced by words in the term document matrix due to certain lexical properties (synonymy and polysemy).

Other Term Extraction Based Methods

Colace *et al.* (2014) demonstrate that the feature set, based on weighted pairs of words, is capable of overcoming the limitations of simple structures when there is only a small number of labelled documents. A new linear single label supervised classifier is proposed that achieved a better performance than existing methods when the size of the training set is about 1% of the original and composed of only positive examples. The proposed feature set is extracted from the corpus using a global method for term extraction based on the Latent Dirichlet Allocation Blei *et al.* (2003) implemented as the Probabilistic Topic Model Griffiths *et al.* (2007).

This thesis focuses on the development of the novel term weighting method which uses only statistical information and no stop word or ignore word filtering. Stemming is also not required for the approaches developed in this study, which makes them independent from the rules of particular language and specific vocabulary of the domain. The term weighting technique introduced in this work uses information about the class labels and, therefore, belongs to the field of supervised term weighting. The dimensionality reduction algorithm

proposed in this thesis is based on the combination of term clustering (unsupervised method) and optimization of the obtained term clusters weights (supervised method). It belongs to the dimensionality reduction algorithms by feature extraction. Another dimensionality reduction algorithm implemented in this thesis is based on the term filtering approach. In this method given a text preprocessing and a filter level the reduced feature space is obtained by excluding the terms with weights that are less than the filter level. This approach belongs to the dimensionality reduction algorithms by feature selection.

2.2 State of the Art in Supervised Classification Methods

2.2.1 Introduction

Supervised classification can be defined as a task to choose using the class annotations of the training data to which category the given test object belongs. Text classification includes an additional step of text preprocessing, which can influence significantly on the performance of the classification algorithm.

With the rise of blogs, social networks and e-commerce sites, there is a great interest in the fields of supervised and semi-supervised text classification with the focus on revealing user sentiments and opinions (Eirinaki *et al.* (2012) and Palus *et al.* (2011)), on discovering and classification the health service information obtained from the digital health ecosystems (Dong *et al.* (2011) and Karavasilis *et al.* (2010)) and on the classification of web resources for improving the quality of web searches (Iqbal & Shah (2012), Grzywaczewski & Iqbal (2012), Liu (2009) and Colace *et al.* (2013)).

This section describes the state of the art in supervised learning models with the focus on the application to the text classification. First, the methods of evaluation of the text classification system are presented. Then, some well known classification algorithms are discussed, which can be applied for text categorization.

2.2.2 Classification Algorithms

Almost all machine learning algorithms can be applied to the text classification task. The most common and well performed methods are Support Vector Machine, Bayes Classifier, k-nearest neighbors algorithm and artificial neural networks.

A broad survey of a wide variety of text classification methods may be found in Sebastiani (2002), Yang & Liu (1999) and Aggarwal & Zhai (2012) and several of the discussed techniques have also been implemented and are publicly available through multiple toolkits such as the BoW toolkit (McCallum (1996)), Mallot (McCallum (2002)), WEKA and LingPipe.

2.2.2.1 Linear Classifiers

Linear classifiers try to identify the class of objects based on the value of a linear combination of feature vector components.

Naive Bayes Classifier

Bayesian or probabilistic classifiers Lewis (1998) have been widely used for text categorization. They use the joint probabilities of words and classes to estimate the probabilities of each class for an input document.

Let n be a set of document vectors $D = \{d_1, \ldots, d_n\}$, which are classified into a set C of m classes, $C = \{c_1, \ldots, c_m\}$. Bayesian classifiers estimate the probabilities of each class c_i for an input document d_j as:

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)} \tag{2.29}$$

where $P(d_j)$ is the probability that the randomly taken document will be equal to d_j in the feature space; $P(c_i)$ is the probability that the randomly taken document will belong to the class c_i. Since the number of possible documents d_j is very high, it is difficult to estimate $P(d_j|c_i)$.

To simplify the estimation of $P(d_j|c_i)$, Naive Bayes assumes that the probability of a given word or term is independent of other terms that appear in the same document. This may seem as an over simplification, but in fact Naive Bayes presents results that are very competitive with those obtained by more elaborate methods. Moreover, because only words and not combinations of words are used as predictors, this naive simplification allows the computation of the model of the data associated with this method to be far more efficient than other non-naive Bayesian approaches. Using this simplification, it is possible to determine $P(d_j|c_i)$ as the product of the probabilities of each term that appears in the document. So, $P(d_j|c_i)$, where $d_j = (w_{1j}, \ldots, w_{|T|j})$, may be estimated as

$$P(d_j|c_i) = \prod_{k=1}^{|T|} P(w_{kj}|c_i) \tag{2.30}$$

where $|T|$ is the number of terms in d_j document.

The document belongs to the class which has the higher probability of $P(c_i|d_j)$:

$$\text{class of } d_j = \arg\max_{c_i} P(c_i|d_j) \tag{2.31}$$

In Chen *et al.* (2009) Naive Bayes classifier is used to present two feature evaluation metrics applied on multi-class text databases: Multi-class Odds Ratio (MOR), and Class

Discriminating Measure (CDM). Since Naive Bayes classifiers are very simple, efficient and highly sensitive to feature selection, i.e. the choice of features is significant for them. The results indicate that the proposed CDM and MOR gain obviously better selecting effect than other feature selection approaches.

Numerical experiments conducted in Meena & Chandran (2009) have shown an improvement of the performance of the Naive Bayes classifier trained only on positive features selected by statistics based method (CHIR) in comparison with other standard classification methods for the 20 Newsgroup benchmark.

In the work Ganiz *et al.* (2011) a new supervised classification algorithm is proposed. It is called Higher Order Naive Bayes (HONB) and it leverages higher order relations between features across different instances. Experimental results on several benchmark corpora show that the introduced approach achieves significantly better classification accuracy over the baseline methods.

Perceptron

Perceptron has been first proposed by Rosenblatt (1958). Generally perceptron is defined as a binary classifier that can be naturally generalized into a multiclass classifier. It maps the input vector $x \in \mathbb{R}^n$ to the binary output $f(x) \in \{0, 1\}$.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{2.32}$$

where $w \in \mathbb{R}^n$ is a vector of weights and b is a constant value; $w \cdot x$ is the scalar product of the input vector x and weights vector w.

The learning algorithm of the perceptron modifies the weights vector w and value b in order to separate the training sample by a hyperplane.

As all linear classifiers perceptron cannot classify correctly all input vectors if the training set is not linearly separable which means that there is no hyperplane which can divide the feature space into positive and negative examples. However, the perceptron is guaranteed to converge (but the number of iteration it will need to converge is generally unlimited) if there is such a hyperplane.

Authors of Ng *et al.* (1997) describe the text categorization algorithm based on perceptron learning with a new feature selection metric called correlation learning. The usability of the proposed learning approach has been compared with rule based expert system that uses a text categorization shell built by Carnegie Group, however the automated approach gives a lower accuracy and the semi-automated method produces the results which are close to the rule based approach.

In the work Liu *et al.* (2013) a stochastic perceptron algorithm in the large margin framework is proposed. A stochastic choice procedure chooses the direction of next iteration. Authors of Liu *et al.* (2013) have proved that with the finite number of iterations the stochastic perceptron finds a suboptimal solution with high probability if the input examples are separable. For large-scale high-dimensional data, the kernel version of the stochastic perceptron should be used to reduce the required memory space. The experimental results conducted in Liu *et al.* (2013) show that this kernel stochastic perceptron achieves almost the same accuracy as the contemporary algorithms on the real-world data sets, however it requires much less CPU running time.

2.2.2.2 Support Vector Machines

The Support Vector Machine has been proposed by Cortes & Vapnik (1995) and has been first applied to text classification by Joachims (1998).

The SVM is a method for training linear classifiers. It is based on statistical learning algorithms: it maps the documents into the feature space and attempts to find a hyperplane that separates the classes with largest margins. The SVM can be interpreted as an extension of the perceptron. It simultaneously minimizes the empirical classification error and maximizes the geometric margin.

Let $\mathcal{D} = \{(\mathbf{d}_i, y_i) \mid \mathbf{d}_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\}_{i=1}^n$ be training data where $\mathbf{d}_i$ is a real vector; y_i is a class label which is equal to -1 if the$i$th document belongs to the first class and is equal to 0 if it belongs to the second one. SVM algorithm can be considered as an optimization problem:

$$-\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{d}_i, \mathbf{d}_j) \rightarrow \min \tag{2.33}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } \forall i\ \alpha_i \geq 0, \tag{2.34}$$

where $(\mathbf{d}_i, \mathbf{d}_j)$ is the dot product of the vectors $\mathbf{d}_i$ and $\mathbf{d}_j$.

The set of optimal coefficients α_i^* can then be used to construct the hyperplane that correctly separates all the training examples with the maximum margin:

$$w \cdot \mathbf{d} = \sum_{i=1}^{n} \alpha_i y_i (\mathbf{d}_i \cdot \mathbf{d}_j) \text{ and } b = \tfrac{1}{2}(w \cdot \mathbf{d}_\circ + w \cdot \mathbf{d}_\bullet). \tag{2.35}$$

The Equation 2.35 shows that the obtained weight vector of the hyperplane is constructed as a linear combination of the training objects. This linear combination only includes objects with $\alpha_i > 0$; support vectors that have minimum distance to the hyperplane. Figure 2.3

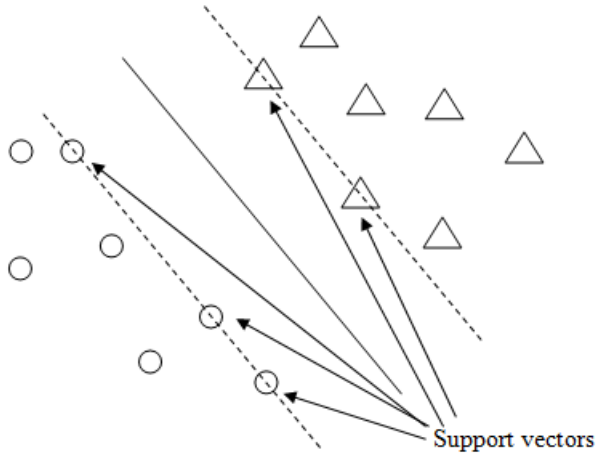

Figure 2.3: Support Vector Machine

illustrates a scheme of the Support Vector Machine.

In the study presented in Rushdi Saleh *et al.* (2011) the SVM is applied for opinion mining tasks with several term weighting schemes. The experiments conducted in Rushdi Saleh *et al.* (2011) have proved the feasibility of the SVM for different domains.

2.2.2.3 Kernel Estimation

Vector Method

The Vector method is one of the oldest methods applied in text classification tasks. In this approach, documents and classes are represented as a set of weighted terms (vectors in a m-dimensional space, where m is the total number of terms). Based on the term weights, every document can be ranked by decreasing order of similarity to the class. The similarity of each document d_j to the class c_i is computed as the cosine of the angle formed by the vectors that represent each of these vectors:

$$\text{cosine\_similarity}(d_j, c_i) = \frac{(d_j, c_i)}{\|d_j\| \, \|c_i\|}. \tag{2.36}$$

The category of the input document is defined as the category of the closest class vector.

k-nearest Neighbor Algorithm

k-nearest-neighbor (k-NN) classification is one of the most fundamental and simple classification methods and it is usually applied when there is little or no prior knowledge about the distribution of the data. An unpublished US Air Force School of Aviation Medicine report by Fix & Hodges Jr (1951) introduced a non-parametric method for pattern classification that

has since become known the k-nearest neighbor rule. For text classification k-nearest-neighbor algorithm has been proposed by Masand *et al.* (1992). The idea of the method is to determine the category of the input document on the basis of not only the document which is the nearest to it in the feature space, but also on the k closest documents. The Vector method can be considered as an instance of the k-NN with $k = 1$.

Weinberger & Saul (2009) present a way to learn a Mahalanobis distance metric for k-NN classification from labelled examples. In this approach, the metric is trained with the goal that the k -nearest neighbors should always belong to the same class and objects from different classes are separated by a large margin. On several data sets of varying size and difficulty, the metrics trained in this way produce significant improvements in k-NN classification.

Nearest Centroid Classifier (Rocchio Classifier)

Rocchio Classifier is one of the simplest classification methods. It has been proposed by Rocchio (1971).

For each category c the weighted centroid is calculated using the following equation 2.37.

$$g_c = \frac{1}{|v_c|} \sum_{d \in v_c} d - \gamma \frac{1}{|\overline{v_{c,k}}|} \sum_{d \in \overline{v_{c,k}}} d, \tag{2.37}$$

where v_c is set of documents that belongs to the category c, $\overline{v_{c,k}}$-k documents that do not belong to the class c and that are close to the centroid $\frac{1}{|v_c|} \sum_{d \in v_c} d$ and γ is parameter of the relative importance of negative precedents consideration.

The application of this method to text classification was proposed by Hull (1994) and it is used in the case where the role of negative precedents is deemphasized (usually $\gamma = 0.25$).

There exist several other methods to calculate the centroids:

1. The average formula where each centroid g_c is represented as an average of all vectors which belong to the class c

$$g_c = \frac{1}{|v_c|} \sum_{d \in v_c} d \tag{2.38}$$

2. The sum formula where each centroid g_c is represented as a sum of all vectors which belong to the class c.

$$g_c = \sum_{d \in v_c} d \tag{2.39}$$

3. The normalized sum formula where each centroid g_c is represented as a sum of all vectors which belong to the class c, normalized so that it has unitary length.

$$g_c = \frac{1}{|\sum_{d \in v_c} d|} \sum_{d \in v_c} d \tag{2.40}$$

The processing document is assigned to the class, whose centroid is the closest to it.

This method has an advantage: weighted centroids can be quickly recalculated after new classified documents are added. This characteristic is useful, for example, in adaptive filtering task when the user can correct system by consequently giving the correct answers. In this case the Rocchio Classifier can improve relatively quickly the classification on each step.

Miao & Kamel (2011) present a pairwise optimized Rocchio algorithm, which dynamically adjusts the prototype position between pairs of categories. Numerical results conducted in Miao & Kamel (2011) have shown the performance improvement over the conventional Rocchio method and competitive results in comparison with Support Vector Machine (SVM) on three benchmark corpora: the 20-Newsgroup, Reuters-21578 and TDT2.

Nearest centroid classifiers are very efficient during the classification stage since time and memory spent are proportional to the number of classes, rather than to the number of training documents as is the case for the Vector method and k-nearest neighbor classifiers.

Distance functions

The performance of k-nearest neighbor algorithm, Vector method and Rocchio Classifier classification depends significantly on the metric used to compute distances between different examples. In order to estimate the distance between the document and the class centroids it is possible to use different metrics. In this thesis in order to calculate the distance between $x = (x_1, x_2, \ldots, x_T)$and $y = (y_1, y_2, \ldots, y_T)$ the following metrics have been applied:

- taxicab metric (Equation 2.41)

$$\text{taxicab}(x, y) = \sum_{i=1}^{T} |x_i - y_i| \tag{2.41}$$

- Euclidean distance (Equation 2.42)

$$\text{euclidean}(x, y) = \sqrt{\sum_{i=1}^{T} (x_i - y_i)^2} \tag{2.42}$$

- cosine similarity (Equation 2.43)

$$\text{cosine\_similarity}(x, y) = 1 - \cos(x, y) = 1 - \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2}} \tag{2.43}$$

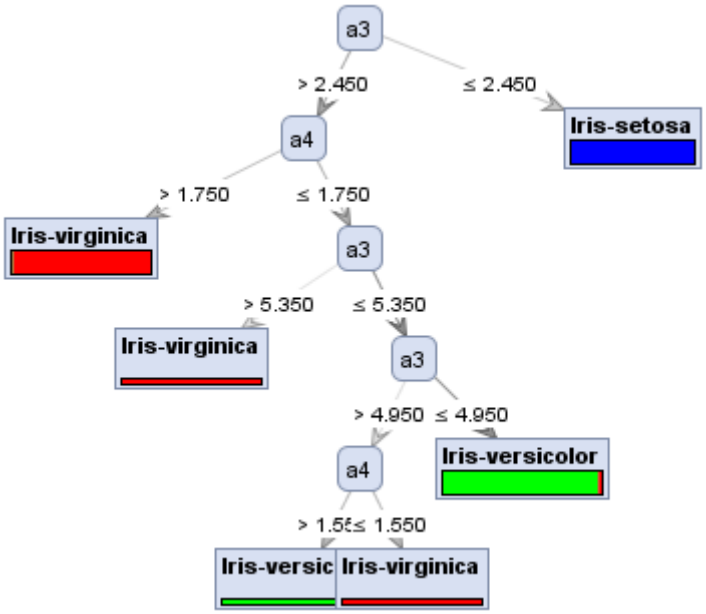

Figure 2.4: Decision tree obtained on Iris database using RapidMiner

- Hamming metric (Equation 2.44)

$$\text{hamming}(x, y) = \sum_{i=1}^{T} \delta(x_i, y_i), \tag{2.44}$$

where $\delta(x_i, y_i)$ is defined as

$$\delta(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i \end{cases}. \tag{2.45}$$

2.2.2.4 Decision Trees

Decision tree model is one of the machine learning methods in which an algorithm is considered as a tree (sequence of connected operations based on comparisons between features and some constant values).

An example of the decision tree obtained using RapidMiner on Iris database from UCI Machine Learning repository.

A question answering system may use classification techniques to improve its performance. Yang & Chua (2004a,b) have proposed to find answers to list questions through web page functional classification. A number of queries are formulated from the list question and processed. The web pages are obtained as a result of the search and then classified by decision trees.

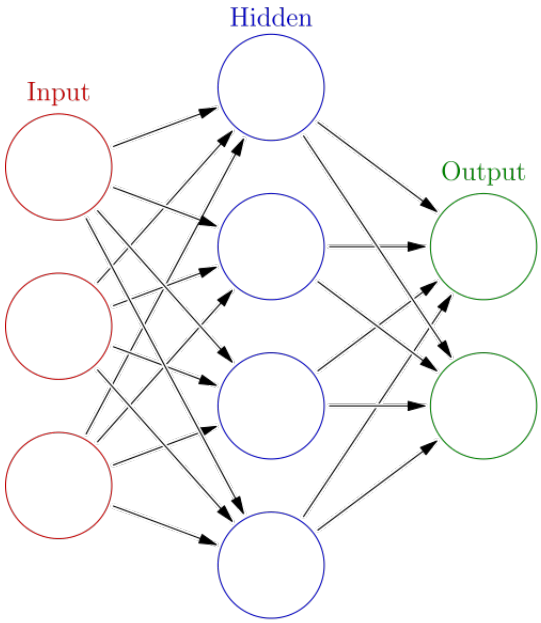

Figure 2.5: Artificial neural network

2.2.2.5 Artificial Neural Networks

Artificial neural networks are computational models that are used in various tasks including machine learning and pattern recognition. This learning model is inspired by human brain and is usually considered as a directed graph with artificial neurons as nodes which can solve problems by signal propagation through the network. Figure 2.5 illustrates the model of artificial neural networks.

In recent years, artificial neural networks have been applied for text classification problems to improve efficiency. Text categorization models using back-propagation neural network (BPNN) and modified back-propagation neural network (MBPNN) are proposed in Yu *et al.* (2008) for documents classification (in the introduced algorithm the feature selection method is applied to reduce the feature space dimensionality and also to improve the classification quality). New Neural network based text classification method has been presented in Trappey *et al.* (2006), which is helpful for companies to manage patent documents more effectively.

In the work of Zhang & Zhou (2006) the neural network algorithm which is called Back propagation for Multi-Label Learning (BP - MLL) has been introduced. It is based on the usual Back propagation algorithm with a novel error function capturing the characteristics of multi-label learning (the class labels which belong to the given object are ranked higher than the class labels which do not belong). The numerical experiments conducted on two applications: functional genomics and text categorization have shown that the BP-MLL performs

better in comparison to some well-established multi-label learning algorithms.

2.2.2.6 Learning Vector Quantization

Learning vector quantization (LVQ) algorithm is a prototype-based supervised classification method. An LVQ system is represented by the set of vectors (prototypes) which are defined in the same feature space as the classifying data. If the number of prototypes equals to the number of classes than LVQ becomes similar to the Rocchio classifier. However, generally each class can have as many prototypes as needed. Algorithm 2.2 presents the main steps of the LVQ.

Algorithm 2.2 Learning vector quantization

1. Randomly create the set of prototypes for all classes.
2. For each object from the the training data find the closest prototype according to a given distance measure.
3. For all train objects assign the class of the chosen prototype.
4. Modify each prototype in the way that it comes closer to the data point if it is correctly classified and moves away from the data point if it is classified incorrectly.

Martín-Valdivia *et al.* (2007) have proposed an alternative to existing text classification models which applies LVQ to the task of text classification and the task of word sense disambiguation. The introduced algorithm is a neural network based on the Kohonen model. Experiments have been carried out using the Reuters-21578 text collection (for text categorization) and the Senseval-3 corpus (for word sense disambiguation).

The study carried out in Umer & Khiyal (2007) evaluates the LVQ for the text classification task. It has shown that LVQ requires less training examples and trains faster than other methods. According to Umer & Khiyal (2007) LVQ outperforms k-NN, Rocchio, Naive Bayes and produces the comparable results to SVM.

In Neto & Barreto (2013) a novel method, called Opposite Maps, is introduced which is used to generate reduced sets for efficiently training of support vector machines. The idea of this method is based on the training two vector quantization algorithms (one for positive examples and one for negative examples) and then finds the closest code vectors obtained from these two vector quantization algorithms. Experimental results have shown the efficiency of the proposed algorithm and its independence of the type of the used vector quantization algorithm.

2.2.2.7 Other Classification Algorithms

In the previous sections the overview of the most popular strategies in machine learning algorithms applied for the text classification has been presented. However, there are of course methods which do not strictly belong to one type of the algorithms or which are significantly different from the ones described in this chapter. Among these, the most interesting ones are

- **Bayesian Inference Networks** in Dumais *et al.* (1998); Lam *et al.* (1997); Tzeras & Hartmann (1993)
  Bayesian network (or belief network) is a statistical model that visualizes a set of random variables and how they depend on each other in directed acyclic graph. Bayesian networks are able to compute a probability that the particular source provided the obtained event.
  The nodes of Bayesian network are considered as random variables (observable quantities, latent variables, unknown parameters or hypotheses). If two nodes are connected with the edge than the random variables which these nodes represent are conditionally dependent of each other.

- **Genetic Programming** in Clack *et al.* (1997); Masand (1994)
  Genetic programming technique proposed by Koza & James (1992) evolves classifying agents. This agent can be a parse-tree representation of a user's particular information need, symbolic expression etc.

- **Maximum Entropy Classifier** in Manning & Schütze (1999)
  Maximum Entropy (multinomial logistic regression, softmax regression) is a classification method that applies logistic regression to the multi-class tasks. Given a set of independent variables this learning model predicts the probabilities of the outputs of a dependent variable which is distributed categorically.

- **Fuzzy Classifier** in Dietterich (1998)
  Fuzzy classification is a process of grouping objects into the so called fuzzy sets. The membership function of the fuzzy set is defined as a real value between 0 and 1, where 0 means that this object does not belong to this fuzzy set and 1 means that it definitely belongs to the set; function values in between define the probability of the object to belong to the fuzzy set.

- **Hidden Markov models** in **Miller *et al.* (1999)**
  Hidden Markov Model is a statistical machine learning model where the optimizing system is assumed to be a Markov process with unobserved states. It can be considered as a simplest dynamic Bayesian network.

The most popular methods for text classification are Support Vector Machine, Artificial Neural Network, k-nearest neighbor algorithm, Bayes and Rocchio classifiers. In order to evaluate the quality of the obtained text preprocessing one has to apply different classifiers, since some classifiers tend to work better in combination with the particular preprocessings. In this thesis various classification algorithms have been applied with different text representations.

Another contribution of this thesis is related to the automatic generation and parameters adjustment of existing classification algorithms (Support Vector Machine and Artificial Neural Network have been selected as the ones with the best performance), since it is not clear how to choose the structure and weight coefficients for the ANN and kernel function and coefficients for the SVM manually.

2.2.3 Performance Measures

A classification system is usually evaluated experimentally rather than analytically. Since to prove that the system is correct and complete the mathematical formulation of the classification problem should be given. That is nonformalizeable because of the subjectivity of whether the document belongs to the certain category. Therefore, the performance of the text classification system is evaluated in terms of how well the system predicts the categories of the input vectors with the knowledge of class labels provided by experts.

2.2.3.1 Accuracy and Error

Accuracy is defined as the ratio between correctly classified documents and all classified documents.

$$\text{accuracy} = \frac{|\{\text{correctly classified documents}\}|}{|\{\text{all documents}\}|} \tag{2.46}$$

Accuracy can be calculated using the following formula in terms of true/false positives (TP and FP) and true/false negatives (TN and FN).

$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.47}$$

Error is defined as a ratio between incorrectly classified documents and all classified documents.

$$\text{error} = \frac{|\{\text{incorrectly classified documents}\}|}{|\{\text{all documents}\}|} = 1 - \text{accuracy} \tag{2.48}$$

2.2.3.2 Precision

Precision is defined as the ratio between correctly classified documents in the class c and all documents which have been assigned to the class c.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \tag{2.49}$$

In the following formula precision is written using true/false positives and true/false negatives (also known as Type I and II errors).

$$\text{precision} = \frac{TP}{TP + FP} \tag{2.50}$$

2.2.3.3 Recall

Precision is defined as the ratio between correctly classified documents in the class c and number of documents which truly belong to the class c.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \tag{2.51}$$

Recall can be also understood as

$$\text{recall} = \frac{TP}{TP + FN} \tag{2.52}$$

2.2.3.4 F-score

A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.53}$$

There are several reasons that the F-score can be criticized in particular circumstances due to its bias as an evaluation metric. This is also known as the F_1 measure, because recall and precision are evenly weighted.

It is a special case of the general F_β measure (for non-negative real values of β):

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{2.54}$$

F-score can be also written in terms of true/false positives and true/false negatives.

$$F_\beta = (1 + \beta^2) \cdot \frac{TP}{(1 + \beta^2) \cdot TP + \beta^2 \cdot FN + FP} \tag{2.55}$$

	Precision $\frac{TP}{TP+FP}$	Recall $\frac{TP}{TP+FN}$	F score $\frac{2 \cdot TP}{2TP+FN+FP}$	Accuracy $\frac{TP+TN}{TP+FP+TN+FN}$
Rejector $TP = FP = 0$	$\frac{0}{0}$	$\frac{0}{FN} = 0$	$\frac{0}{FN} = 0$	$\frac{TN}{TN+FN}$
Acceptor $TN = FN = 0$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP} = 1$	$\frac{2 \cdot TP}{2 \cdot TP+FP}$	$\frac{TP}{TP+FP}$
only positive examples $FP = TN = 0$	$\frac{TP}{TP} = 1$	$\frac{TP}{TP+FN}$	$\frac{2 \cdot TP}{2TP+FN}$	$\frac{TP}{TP+FN}$
only negative examples $FN = TP = 0$	$\frac{0}{FP} = 0$	$\frac{0}{0}$	$\frac{0}{FP} = 0$	$\frac{TN}{FP+TN}$

Table 2.2: Trivial cases in text classification

Precision and recall do not make sense in isolation from each other. Trivial cases in text classification and values of the described performance measures are presented in Table 2.2.

2.2.3.5 Other Measures

Some other measures of the classification quality include:

- effectiveness, training efficiency, classification efficiency proposed in Dumais *et al.* (1998),
- utility proposed in Lewis (1995),
- relative success proposed in Sable & Hatzivassiloglou (2000),
- adjacent score proposed in Larkey (1998),
- coverage proposed in Schapire & Singer (2000),
- one-error proposed in Schapire & Singer (2000),
- Pearson product-moment correlation proposed in Larkey (1998),
- recall at n proposed in Larkey & Croft (1996),
- top candidate proposed in Larkey & Croft (1996),
- top n proposed in Larkey & Croft (1996).

2.2.3.6 Micro- and Macro-Averaging

The performance measures (Precision, Recall and F-score) can be calculated over the whole corpus, which is called micro-averaging , or for each class and then in all classes, which is called macro-averaging .

In micro-averaging, each document contributes the same to the average and, therefore, small classes have a small impact on the final value. In macro-averaging, first the average for each class is calculated, and then the average of each class contributes to the final average. In this case small classes will have the same impact as big classes.

- Micro-Average

$$\text{Precision}= \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i+FP_i)} \tag{2.56}$$

$$\text{Recall}= \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i+FN_i)} \tag{2.57}$$

- Macro-Average

$$\text{Precision}= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{TP_i}{(TP_i+FP_i)}\right) \tag{2.58}$$

$$\text{Recall}= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{TP_i}{(TP_i+FN_i)}\right), \tag{2.59}$$

where TP_i is the number of documents correctly classified to the class i; FP_i is the number of documents which have been incorrectly assigned to the class i; FN_i is the number of documents from the corpus which should have been (but have not been) classified to the class i.

Micro- and macro-averaging can produce significantly different results. This difference becomes particularly important when the corpus is unbalanced, which means that the classes have very different numbers of documents. If one category has just few training documents, then using macro-averaging the performance measure of the classifier will emphasize the importance of detection this small category and using micro-averaging the performance measure will be influenced much less with the classification result obtained on this small category.

2.3 State of the Art in Semi-Supervised Classification Methods

2.3.1 Introduction

In the real-world problems there is often not enough available information about the class annotations to build a sufficient supervised classifier. If there is some data concerning the class label, then it is possible to apply some semi-supervised methods to separate the objects. If

there is no such data, then only unsupervised learning algorithms can be used. The lack of class annotations is usually the case with text classification, since text databases are often extracted from the Internet, where only a few documents are labelled, however, in the unlabelled part of the text collection there is statistical information useful for the classification task. Therefore, semi-supervised and unsupervised methods are particularly important for text classification.

Unsupervised classification (also called clustering) differs from the supervised classification in the way that there is no available class annotations for the training data in the clustering task and the task is to analyze the relations between the objects and find the pattern in the data that can divide them in the most suitable way. The question which partition is better is non-trivial and clustering algorithms use the specially designed criterion to choose how to separate the objects. These unsupervised criteria are often based on the distance matrix between the objects.

This chapter provides a survey of the most popular techniques for unsupervised machine learning (clustering) and their applications. It introduces the main existing approaches in the fields of clustering and semi-supervised learning models.

2.3.2 Clustering Algorithms

Since the 1930's researchers are interested in the field of cluster analysis, which focuses on finding the structure in unlabelled data using the similarities in the data features themselves. This subsection provides an overview of some well known clustering strategies. Connectivity-, centroid-, distribution- and density-based clustering methods are discussed in detail.

2.3.2.1 Connectivity-based Clustering (hierarchical clustering)

Hierarchical clustering is one of the cluster analysis methods that forms a hierarchy of clusters. Strategies for hierarchical clustering can be divided into two types:

- *Agglomerative:* This is a "bottom up" approach. In the beginning each object is considered as a separate cluster and pairs of clusters are combined as one moves up the hierarchy.

- *Divisive:* This is a "top down" approach. In the beginning all objects belong to one cluster and the cluster is separated as one moves down the hierarchy.

In general, the decision which clusters should be united or which cluster should be split is performed according to the situation on each step. The results of hierarchical clustering are usually presented in a dendrogram.

The main idea of the hierarchical clustering is based on the assumption that the objects which are “close” to each other in feature space are somehow similar. These algorithms connect

(agglomerative strategy) or disconnect (divisive strategy) the objects to form clusters based on the distance between them.

Agglomerative Hierarchical Clustering

There are several methods of agglomerative hierarchical clustering. In these methods one starts with small clusters, each of them contains only one element. Then the clusters are united consequentially into larger clusters, until there is only one big cluster, which includes all objects. In every iteration two clusters are combined, which have the shortest distance between them. All hierarchical algorithms are sensitive to the chosen distance metric. The definition of "the shortest distance" between two clusters is what differentiates between the different agglomerative clustering methods.

Let $a_1, a_2, \ldots, a_N$ denote the objects. N is the total number of the objects. Algorithm 2.3 shows the common scheme of agglomerative hierarchical clustering.

Algorithm 2.3 Common scheme of agglomerative hierarchical clustering

1. Start with disjointed objects (each object a_i forms its own cluster c_i). Clustering $\{c_1, c_2, \ldots, c_N\}$, number of iterations $i = 0$.
2. Calculate all distances between pairs of clusters.
3. Find two closest clusters c_i and c_j $(i < j)$.
4. Add cluster c_j to cluster c_i. Increment i, $i = i + 1$.
5. Recalculate the distances between new cluster c_i and other clusters according to the chosen metric. The new clustering is $\{c_1, c_2, \ldots, c_{N-i}\}$.
6. If $(N - i) > 1$ than go to the step 3; else stop.

The algorithm of divisive hierarchical clustering is similar, but a divisive method starts with one cluster that includes all objects and divides it iteratively.

There exist many hierarchical clustering algorithms which differ in the way of measuring the distance between clusters and criterion for deciding which clusters should be united at the current step. The distance functions (linkage criteria) that are most commonly used in the hierarchical clustering and the according clustering algorithms are given in the following.

- *Single-linkage clustering (nearest neighbors clustering)*

The distance $D(X, Y)$ between clusters X and Y is given by the following equation.

$$D(X, Y) = \min_{x \in X, y \in Y} d(x, y), \tag{2.60}$$

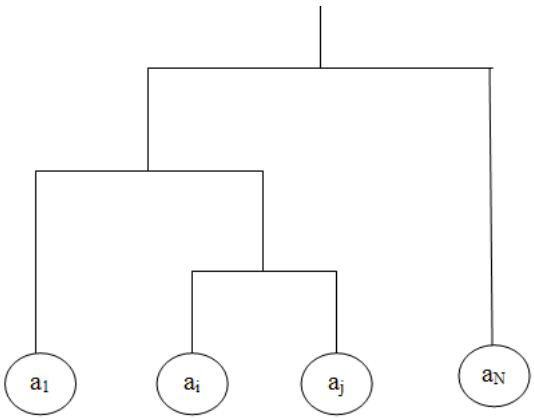

Figure 2.6: Agglomerative dendrogram

where X and Y are any two sets of elements considered as clusters, and $d(x, y)$ denotes the distance between the two elements x and y.

A drawback of this method is the so-called chaining phenomenon, which refers to the gradual growth of a cluster as one element at a time gets added to it. This may lead to clusters which are not homogeneous and problems in finding classes that could usefully subdivide the data.

- *Complete-linkage clustering (farthest neighbor clustering)*

The distance $D(X, Y)$ between clusters X and Y is given by the following equation.

$$D(X, Y) = \max_{x \in X, y \in Y} d(x, y), \tag{2.61}$$

where X and Y are any two sets of elements considered as clusters, and $d(x, y)$ denotes the distance between the two elements $x \in X$ and $y \in Y$.

- *Average-linkage clustering*

The distance $D(X, Y)$ between clusters X and Y is given by the following equation.

$$D(X, Y) = \frac{1}{|X| \cdot |Y|} \sum_{x \in X} \sum_{y \in Y} d(x, y), \tag{2.62}$$

where X and Y are any two sets of elements considered as clusters, $|X|$ is the number of elements in cluster X, $|Y|$ is the number of elements in cluster Y and $d(x, y)$ denotes the distance between the two elements $x \in X$ and $y \in Y$.

- *Ward's method clustering (Ward's minimum variance criterion)*

Ward's minimum variance criterion Ward Jr (1963) aims to minimize the total variance within a cluster. The pair of clusters with minimum distance between them are united at each step. This pair of clusters is united if it provides the minimum increase (a weighted squared distance between cluster centres) in the total variance after combining them in one cluster.

The distance $D(X, Y)$ between initial clusters X and Y is given by the following equation.

$$D(X, Y) = \|x_c - y_c\|^2, \tag{2.63}$$

where X and Y are any two sets of elements considered as clusters, x_c is a center of cluster X, y_c is a center of cluster Y; and $\|x - y\|$ denotes the Euclidean distance between the two elements $x \in X$ and $y \in Y$.

- *Centroid linkage clustering*

The distance $D(X, Y)$ between clusters X and Y is given by the following equation.

$$D(X, Y) = \|c_x - c_Y\|, \tag{2.64}$$

where X and Y are any two sets of elements considered as clusters, c_x and c_Y are centroids of the clusters X and Y respectively.

- *Minimum energy clustering*

The distance $D(X, Y)$ between clusters X and Y is given by the following equation.

$$D(X, Y) = \frac{1}{|X| \cdot |Y|} \sum_{x \in X} \sum_{y \in Y} \|x_i - y_j\|_2 - \frac{1}{|X|^2} \sum_{x \in X} \sum_{x \in X} \|x_i - x_j\|_2 - \frac{1}{|Y|^2} \sum_{y \in Y} \sum_{y \in Y} \|y_i - y_j\|_2, \tag{2.65}$$

where X and Y are any two sets of elements considered as clusters, $|X|$ is the number of elements in cluster X, $|Y|$ is the number of elements in cluster Y.

Other linkage criteria include:

- The sum of all intra-cluster variance.
- The probability that candidate clusters spawn from the same distribution function (V-linkage).
- The product of in-degree and out-degree on a k-nearest-neighbor graph (graph degree linkage).
- The increment of some cluster descriptor (i.e., a quantity defined for measuring the quality of a cluster) after merging two clusters

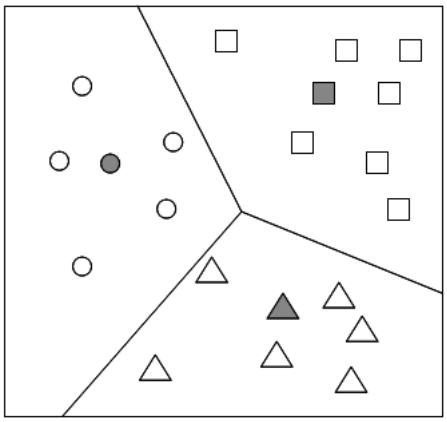

Figure 2.7: k-means clustering

2.3.2.2 Centroid-based Clustering

In this clustering approach, clusters are represented by a central vector (the prototype of the cluster), which may or may not be a member of the database. This is a NP-hard problem and there exist several heuristic algorithms that are able to quickly converge to a local optimum.

- *k-means Clustering*

Given a set of objects $(d_1, \ldots, d_n)$, where each object is a vector in $\mathbb{R}^m$ space, k-means clustering algorithm aims to divide these n objects into k sets $(k \leq n)$ $S = S_1, \ldots, S_k$ in the way that minimizes the within-cluster sum of squares:

$$\min_{S} \sum_{i=1}^{k} \sum_{d_j \in S_i} \|d_j - \mu_i\|^2, \tag{2.66}$$

where μ_i is the mean of vectors in S_i.

Algorithm 2.4 presents the scheme of the k-means clustering.

- *Self-Organizing Map*

A self-organizing map consists of components called nodes or neurons. A weight vector of the same dimension as the input object vectors (a position in the map space) is assigned to each node. The two-dimensional map with a hexagonal or rectangular grid is usually used because two dimensions can be easily visualized. The self-organizing map describes a mapping from a higher-dimensional input space to a lower dimensional map space. An input data vector is placed on the map by finding the node with the nearest (according to the chosen distance metric) weight vector to the data vector.

Algorithm 2.4 k-means clustering

1. Set iteration number $i = 0$.
2. Randomly initialize the vector$A^0 = (a_1, \ldots, a_n)$, where n is the number of objects, $a_i \in \{1, \ldots, k\}$ and k is the number of clusters.
3. Compute similarity matrix $W = DD^T$, where D is a matrix of n objects.
4. $i = i + 1$
5. Calculate squared distance between objects and centroids using equation 2.66.
6. Compute A^i by assigning each object to the class of the nearest centroid.
7. If $A^i = A^{i-1}$ then END else go to the step 4.

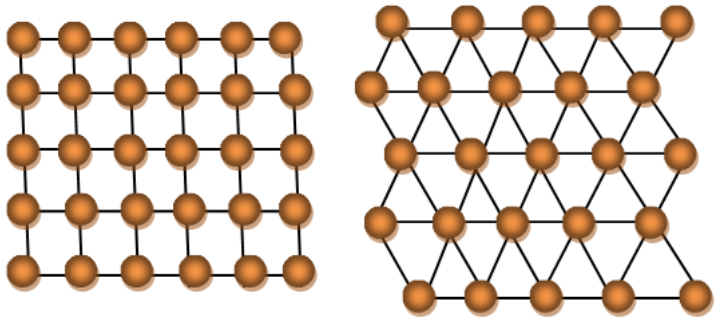

Figure 2.8: Self-Organizing Map

2.3.2.3 Distribution-based Clustering

This clustering model is closely related to statistics and is based on distribution models. Clusters are considered as objects which most likely belong to the same distribution.

The major drawback of these methods is the overfitting problem, if there are no constraints on the model complexity. A more complex model will fit the data better, therefore, more and more complex models can be picked indefinitely.

Gaussian mixture model

It is one well known example of the distribution-based clustering. The data is considered to belong to the fixed set (to overcome the overfitting problem described above) of Gaussian distributions. In the beginning the parameters of these distributions are initialized randomly and are being optimized during the learning process to fit better to the given data.

One of the advantages of this algorithm is that it produces for each object the possibility that this belongs to the certain cluster, therefore if hard cluster set is required than the most

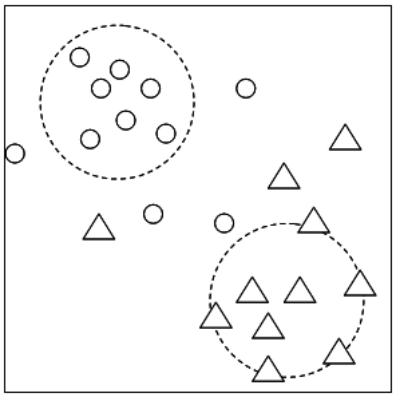

Figure 2.9: Density-based clustering

likely Gaussian distribution is chosen, otherwise it is not necessary.

Another advantage of this clustering method is that it provides not only partition of objects but also the data model which shows the relations between clusters and features.

The drawback of the distribution based models is that the user has to choose the model and it is not a trivial task for the real-world data.

2.3.2.4 Density-based Clustering

In density-based clustering, clusters are considered as areas of higher density than the rest of the feature space. Objects in these sparse areas - that are required to separate clusters - are usually considered to be noise and border points.

2.3.2.5 Other Clustering Algorithms

In the previous sections the overview of the most popular strategies in unsupervised machine learning algorithms has been presented. However, there are of course methods which do not strictly belong to one type of the algorithms or which are significantly different from the ones described in this chapter. Among these, the most interesting one is the graph-based clustering.

Graph-based Clustering

In graph-based clustering, the objects are considered to be nodes of the graph with a set of edges. There are different clustering algorithms based on this model: Metis, Hmetis Karypis & Kumar (1998), mincut Feng *et al.* (2010), Pole-Based Overlapping algorithm Cleuziou *et al.* (2004).

2.3.3 Evaluation Methods

Evaluation of the partitions obtained by unsupervised models is a more difficult task than the evaluation of supervised classifiers, because it is not clear how to interpret the obtained cluster structure. However, some strategies have been proposed in the literature for processing the outcome of a clustering algorithm (it is also called cluster validation). If at least some of the true labels are known for the clustered objects, it is possible to map clusters obtained by the algorithm into the sets of labelled data (external criterion). If true labels are not available, than the base of such an estimation can only be the obtained clusters themselves which means that usually internal criteria are based on the distances between the clustered objects.

2.3.3.1 Internal Evaluation Criteria

Internal evaluation criteria use only the information in the clustered data. They measure the similarity in every cluster and between clusters: clustering algorithms are usually considered as good ones if they have high intra-cluster similarity (it is usually measured as an average distance within the cluster) and low inter-cluster similarity (it is usually calculated as a distance between different clusters). The most frequently used measures are the Davies-Bouldin index and the Dunn index.

- The *Davies-Bouldin index* is given by

$$DaviesBouldin = \frac{1}{n}\sum_{i=1}^{n}\max_{i\neq j}\left(\frac{s_i + s_j}{d(c_i, c_j)}\right), \tag{2.67}$$

where n is the number of clusters; c_i is the centroid of cluster i; s_i is the average distance between all elements in cluster i and centroid c_i; $d(c_i, c_j)$ is the distance between centroids c_i and c_j.

One can see that the clustering partition with small distances between elements in one cluster has high intra-cluster similarity and large distances between elements from different clusters has low inter-cluster similarity. Therefore, the clustering algorithm which provides partition with a lowest Davies–Bouldin index is considered to be the best method according to this criterion.

- The *Dunn index* is calculated as

$$Dunn = \min_{1\leq i\leq n}\left\{\min_{1\leq j\leq n, i\neq j}\left\{\frac{d(i,j)}{\max_{1\leq k\leq n} d'(k)}\right\}\right\}, \tag{2.68}$$

where $d(i, j)$ is the distance between clusters i and j. $d(i, j)$ can be measured using different distance metrics, such as the distance between cluster centroids. $d'(k)$ is the distance between elements in cluster k. $d'(k)$ can be calculated as a maximal distance between elements from cluster k or any other metric.

	true H_0	false H_0
predicted true H_0	correct outcome True Positive (TP)	Type I error False Positive (FP)
predicted false H_0	Type II error False Negative (FN)	correct outcome True Negative (TN)

Table 2.3: Relations between all possible outcomes of the clustering algorithm and the null hypothesis

This criterion tends to find well separated clusters with high density. According to the Dunn index, clustering algorithms which produce clusters with high Dunn index are considered as better methods.

2.3.3.2 External Evaluation Criteria

External criteria are based on the data that has not been used by a clustering algorithm such as a set of pre-classified objects (usually created manually by experts). These evaluation criteria measure how good the obtained set of clusters matches the predetermined set of classes. The measure is usually done in terms of true/false positive and true/false negative outcomes (or types I and II errors). Tabularised relations between truth/falseness of the null hypothesis H_0 and outcomes of the algorithm are presented in Table 2.3.

- Rand measure

As all external evaluation criteria the Rand index shows the similarity between the set of clusters and classification given by experts. The Rand index can be considered as a percentage measure of correct decisions made by the clustering algorithm. It can be computed using the following formula:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.69}$$

One issue with the Rand index is that false positives and false negatives are equally weighted. This may be an undesirable characteristic in some clustering applications. The F-measure addresses this concern.

- F-measure

The F-measure can be used to balance the contribution of false negatives by weighting recall through a parameter $\beta \geq 0$.

Precision is defined as

$$P = \frac{TP}{TP + FP}. \tag{2.70}$$

Recall is given by

$$R = \frac{TP}{TP + FN}. \tag{2.71}$$

F-measure is calculated using the following formula.

$$F_\beta = \frac{(\beta^2 + 1) \cdot P \cdot R}{\beta^2 \cdot P + R} \tag{2.72}$$

Notice that when $\beta = 0$, $F_0 = P$. In other words, recall has no impact on the F-measure when $\beta = 0$, and increasing β allocates an increasing amount of weight to recall in the final F-measure.

- Jaccard index

The Jaccard index is used to quantify the similarity between two datasets. The Jaccard index takes on a value between 0 and 1. An index of 1 means that the two dataset are identical, and an index of 0 indicates that the datasets have no common elements. The Jaccard index is defined by the following formula

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN} \tag{2.73}$$

- Fowlkes-Mallows index

The Fowlkes-Mallows index (the FM index) computes the similarity between the clusters returned by the clustering algorithm and the benchmark classifications. The higher the value of the Fowlkes-Mallows index the more similar the clusters and the benchmark classifications are. It can be computed using the following formula

$$FM = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}} \tag{2.74}$$

The FM index is the geometric mean of the precision and recall P and R, while the F-measure is their harmonic mean. Moreover, precision and recall are also known as Wallace's indices B^I and B^{II} Wallace (1983).

- Other external criteria

  Confusion matrix (also called confidence table, error matrix or matching matrix) where columns represent the outcomes of the algorithm and rows are considered as objects with known class labels.

  Mutual Information shows how much information the obtained clustering and known classification share. Mutual Information is calculated using the following equation

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right), \tag{2.75}$$

where X is the clustering obtained by unsupervised learning model; Y is the classification given by experts; $p(x,y)$ is the joint probability distribution function of X and Y; $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively.

2.3.4 Semi-Supervised Text Classification

Semi-supervised classification is the set of methods that help to improve the performance of the supervised algorithms by the use of both labelled and unlabelled data and by the application of clustering.

There are different approaches to semi-supervised classification which have been introduced in the literature. These include co-training Maeireizo *et al.* (2004), self-training Yarowsky (1995) or generative models Nigam *et al.* (2000). Two detailed surveys on semi-supervised classification have been presented in Zhu (2006) and Seeger *et al.* (2001). Some frequently used semi-supervised methods are: Expectation-Maximization algorithm with generative mixture models, self-training, co-training, transductive support vector machines, and graph-based methods.

2.3.4.1 Self-Training

In this approach to semi-supervised classification, the classifier is trained on the growing set of labelled data starting from the small existing set of labelled objects and at each interaction adding some objects to this set. Note the classifier uses its own predictions to teach itself.

Algorithm 2.5 provides the scheme of self-training approach to semi-supervised classification.

Algorithm 2.5 Self-training approach to semi-supervised classification

1. L_0 is a labelled data; U_0 is unlabelled data; set the iteration $i = 0$.
2. Train the classifier on L_i and apply the model to U_i.
3. Sort the classified objects from U_i according to their confidence scores.
4. Remove the set of the top most confident objects C_i from U_i and add C_i to the labelled data: $U_{i+1} = U_i \setminus C_i$, $L_{i+1} = L_i + C_i$.
5. Set $i = i + 1$ and go to the step 2.

Self-training has been applied to various natural language processing tasks. For example, Yarowsky (1995) has used self-training for word sense disambiguation (the choice of the meaning of the polysemous word given the context: for example the word "set" has over 120 meanings in English). Riloff *et al.* (2003) identifies subjective nouns with the self-training model. Self-training has also been applied to parsing and machine translation. Rosenberg *et al.* (2005) compared the performance of the semi-supervised self-training system for image detection against the performance of the state-of-the-art detector.

2.3.4.2 Co-Training

Co-training Blum & Mitchell (1998); Mitchell (1999) assumes that the feature set can be split into two subsets, where each subset of features is sufficient to train a good classifier and these two sets are conditionally independent on given the category. In this model each classifier uses the predictions of another classifier to teach itself.

Algorithm 2.6 presents the scheme of co-training approach to semi-supervised classification.

Algorithm 2.6 Co-training approach to semi-supervised classification

1. $L_0 = L_0^1 = L_0^2$ is a labelled data; $U_0 = U_0^1 = U_0^2$ is unlabelled data; set the iteration $i = 0$; $F = F_1 + F_2$ is a feature set that consisted of the subsets F_1 and F_2; M_1 is the first classifier that uses the feature subset F_1 and M_2 is the second classifier that uses the feature subset F_2.
2. M_1 and M_2 are trained on L_i^1 and L_i^2 respectively and predict the labels of U_i^1 and U_i^2.
3. Sort the classified objects from U_i^1 and U_i^2 according to their confidence scores.
4. Few examples from the U_i^1 with the highest confidence scores are moved from the U_i^1 to the L_i^2: $U_{i+1}^1 = U_i^1 \setminus C_i^1$, $L_{i+1}^2 = L_i^2 + C_i^1$.
5. Few examples from the U_i^2 with the highest confidence scores are moved from the U_i^2 to the L_i^1: $U_{i+1}^2 = U_i^2 \setminus C_i^2$, $L_{i+1}^1 = L_i^1 + C_i^2$.
6. Set $i = i + 1$ and go to the step 2.

Nigam & Ghani (2000) have compared co-training with the generative mixture models and Expectation-Maximization algorithm (EM) showed that the co-training algorithm performs well if there is conditional independence of the feature subsets and that it is better to give probabilistic membership to the unlabelled data instead of taking only few top objects (this approach is called co-EM). In the case when the feature set cannot be split naturally, Nigam and Ghani have artificially divided the feature set into two subsets randomly.

Jones Jones (2005) has used co-training, co-EM and other related methods to extract

information from text. Zhou et al. Zhou *et al.* (2007) have applied a co-training algorithm using Canonical Correlation Analysis with only one labelled object.

Co-training requires strong assumptions on the splitting of features. There are several works Goldman & Zhou (2000); Chawla & Karakoulas (2005); Balcan *et al.* (2004) which attempted to relax these conditions.

2.3.4.3 Generative Models

Generative models belong to the most well-known semi-supervised classification method which assumes that

$$p(x, y) = p(y) \cdot p(x|y), \tag{2.76}$$

where $p(x|y)$ is an identifiable mixture distribution (i.e. *Gaussian mixture model* is used in many applications).

When the number of unlabelled objects is large, these mixture components can be calculated and in ideal situation only one labelled instance for each category is required to fully determine the distribution. Mixture components can be considered as "soft clusters".

Nigam *et al.* (2000) have applied the Expectation-Maximization algorithm on mixture of multinomials for the task of text classification. Numerical experiments conducted in Nigam *et al.* (2000) have shown that the resulting classifiers provide better results than the classifiers trained only on the labelled data. Fujino *et al.* (2005) proposed to include a "bias correction" term and discriminative training using the maximum entropy principle in order to extend the generative mixture models.

Other strategies called *cluster-and-label* methods attempt to generate the mixture model by means of clustering. For example, in Demiriz *et al.* (1999) a semi-supervised algorithm combines the benefits of the supervised and unsupervised learning methods, where a genetic k-means clustering is generated with a genetic algorithm. The proposed algorithm aims to find a set of k cluster centres which provides an optimum of an internal quality objective and an external criterion that uses available labels. The simultaneous minimization of the linear combination of the supervised and unsupervised criteria is performed.

$$\alpha \cdot ClusteringCriterion + \beta \cdot ClassificationCriterion \rightarrow \min, \tag{2.77}$$

where α and $\beta \in (0, 1)$ are real-valued parameters.

Another approach is to transform the data into a feature representation determined by the generative model. Then the standard discriminative classifier is trained on this new representation. Holub *et al.* (2005) has applied this algorithm for image categorization. First a generative mixture model is trained using one component per class, but instead of directly

using the generative model for classification, each labelled example is converted into a fixed-length Fisher score vector (the derivatives of log likelihood). These Fisher score vectors are then used to train a discriminative classifier such as an SVM.

These semi-supervised methods differ in the way they define the objective function. The target function is usually defined as a linear combination of supervised and unsupervised criteria as in the equation 2.77. In this thesis, however, the training and test examples are not used simultaneously. First, the text representation is built based on the class labels of the training data. In the next step, the terms that "point" to the same category are combined, which means that on this stage the class labels are also used. Then, the hierarchical agglomerative clustering for each group of terms is applied. On this step there is no use of the class annotations. After that, the class labels are used again in the optimization process when the term clusters weights are adjusted to produce better representation with respect to the given task. In order to improve performance for some databases the objective function includes unsupervised criterion in the similar way as in the equation 2.77.

2.4 Optimization Algorithms

Most text classification algorithms involve a careful choice of parameters, which is a complicated task that requires a lot of time if it is done manually by experts. Therefore, parameters of classification methods are usually selected using an optimization method. Since it is unknown how the objective function changes depending on the parameters' values and it is algorithmically defined, various heuristic optimization approaches are used instead of deterministic ones. In this section stochastic optimization algorithms applied in this thesis in order to find the structure and parameters of text classifiers are described. First, the standard genetic algorithm and its parameters are discussed and, then, the metaheuristic approaches for real-valued optimization (original COBRA), for binary-valued optimization (binary COBRA) and for constrained optimization are provided (constrained COBRA).

2.4.1 Standard Genetic Algorithm

This section explains the scheme of the standard genetic algorithm (GA) and provides the definitions of the parameters of GA.

Genetic algorithms are based on the classic view of a chromosome as a string of genes. Using this view Fisher (1958) provided mathematical formula specifying the rate at which particular genes would spread through a population. Genetic algorithms are well-known and widely used methods of global optimization since they can work with algorithmically defined criteria on the binary, real, nominal, mixed etc. data types; they search in parallel in different regions of the feature space. These algorithms of stochastic optimization are inspired by

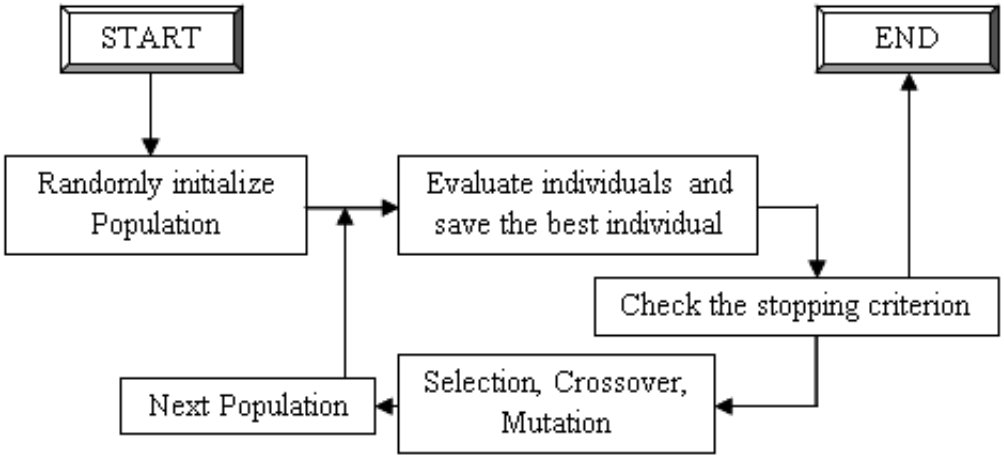

Figure 2.10: Scheme of standard genetic algorithm

the evolution process. This evolutionary principle forms the basis for genetic algorithms. The environment is modelled by the optimization criterion (objective function), while the individuals are represented by their chromosomes.

The basic scheme of the standard genetic algorithm is shown in Figure 2.10.

Standard genetic algorithm is shown in Algorithm 2.7.

Algorithm 2.7 Standard genetic algorithm

1. Randomly initialize the population (usually the candidate solutions, individuals, are encoded as binary sequences). Set generation counter $g = 0$.
2. Calculate the fitness functions (chosen optimization criterion) for all individuals.
3. Save the individual with the best value of fitness function
4. Check if the generation counter is greater than some predefined value: if $g > G$ than END; otherwise go to step 5.
5. Using GA operators: selection, crossover and mutation, form the next population.
6. Increment the generation counter $g = g + 1$. Go to step 2.

Selection, crossover and mutation operators are explained in detail in the following paragraphs.

Selection

Selection is the stage of a genetic algorithm in which individual genomes are chosen from a population for later breeding (recombination or crossover). This stage of GA should satisfy

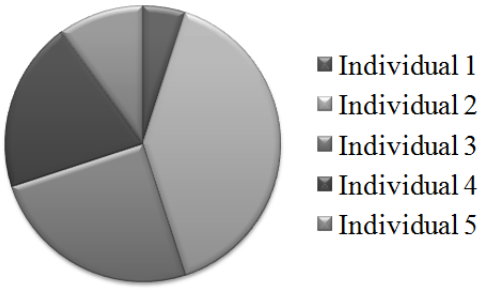

Figure 2.11: Roulette wheel selection

two contradictory factors:

1. Selection should choose individuals with high value of the fitness function
2. Individual with low fitness functions should also have a chance to be selected; otherwise the diversity of the population will become poor and the algorithm will go to the stagnation (the stage when algorithm cannot find better solutions and all population consists of similar individuals)

There are many variants of the selection operator. In the following, the most popular selection types are described.

- *Proportional selection*
  Proportional selection is also known as roulete wheel selection
  The possibility to be chosen for reproduction is calculated by the following formula

$$p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j}, \tag{2.78}$$

  where f_i is a fitness function of the ith individual.

  This variant of selection has a lot of drawbacks.

- *Linear Ranking selection*
  Ranking selection has been proposed by Baker (1985) in order to overcome the drawbacks of the proportionate selection described previously. For ranking selection the population is sorted according to the fitness functions of the individuals and then the rank N is assigned to the best individual and the rank 1 to the worst individual.
  The probability of being selected is calculated as:

$$p_i = \frac{1}{N} = \left(n^- + (n^+ - n^-)\frac{i-1}{N-1} \right), 1 \leq i \leq N, \tag{2.79}$$

  where $\frac{n^-}{N}$ is the probability that the worst individual will be selected and $\frac{n^+}{N}$ is the probability that the best individual will be chosen.

- *Tournament selection*
  Tournament selection is a method of selecting an individual from a population of individuals in a genetic algorithm. Tournament selection involves running several "tournaments" among a few individuals chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for crossover. Selection pressure is easily adjusted by changing the tournament size. If the tournament size is larger, weak individuals have a smaller chance to be selected.
  Algorithm 2.8 presents the scheme of the tournament selection.

Algorithm 2.8 Tournament selection

1. Choose k (the tournament size) individuals from the population at random .
2. Calculate the fitness function of all of them.
3. Choose the best individual from pool/tournament with probability p.
4. Choose the second best individual with probability $p * (1 - p)$.
5. Choose the third best individual with probability $p * ((1 - p)^2)$ and etc.
6. Deterministic tournament selection selects the best individual (when $p = 1$) in any tournament (in the algorithms developed in this thesis only deterministic tournament with the $size = 3$ is used). The chosen individual can be removed from the population that the selection is made from if desired, otherwise individuals can be selected more than once for the next generation.Tournament selection has several benefits: it is efficient to code, works on parallel architectures and allows the selection pressure to be easily adjusted.

There exist other selection types such as:

- *Truncation Selection*
  Using trancation selection only the set of several best individuals is selected and then they all are chosen for further breeding with the same probability.

- *Exponential Ranking Selection*
  The difference between linear variant is that the probabilities of the individuals being chosen are exponentially weighted.

Crossover

Crossover is a stage of genetic algorithms when selected individuals share their genetic material to produce a next population presumably with better fitness functions of individuals.

Several well-known types of recombination operators are shown below.

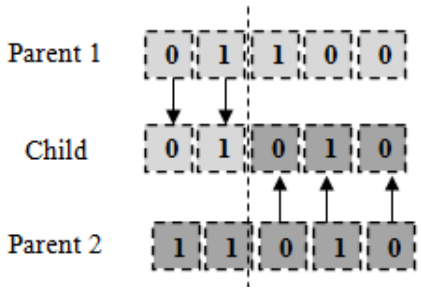

Figure 2.12: One point crossover

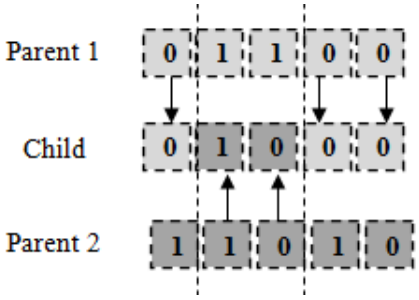

Figure 2.13: Two point crossover

- *One point crossover*
  In this type of cross-over, a point inside the individual's chromosome is randomly selected. The genes of the two parents are divided by this point and are swapped to form the children. Figure 2.12 illustrates one point crossover.

- *Two point crossover*
  Two points inside the individual's chromosome are randomly selected and their genes are divided into parts and swapped between each two consecutive points. Figure 2.13 illustrates one point crossover.

One point and two points crossovers can be generalized into k point crossover, where k points inside the chromosome are selected and the genetic material is swapped between each two consecutive points.

- *Uniform crossover*
  The Uniform Crossover uses a fixed mixing ratio between two parents. Unlike one- and two-point crossovers, the Uniform Crossover enables the parent chromosomes to contribute the gene level rather than the segment level.
  If the mixing ratio is 0.5, the offspring has approximately half of the genes from first

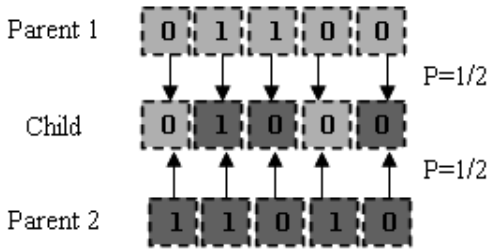

Figure 2.14: Uniform crossover

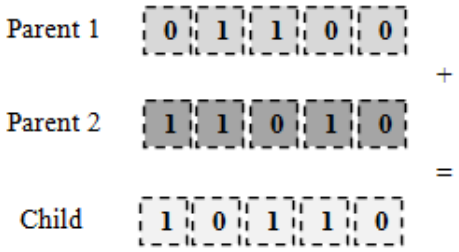

Figure 2.15: Arithmetic crossover

parent and the other half from second parent, although cross over points can be randomly chosen: Figure 2.14 illustrates one point crossover.

The Uniform Crossover evaluates each bit in the parent strings for exchange with a probability of 0.5. Even though the uniform crossover is a poor method, empirical evidence suggest that it is a more exploratory approach to crossover than the traditional exploitative approach that maintains longer schemata. This results in a more complete search of the design space with maintaining the exchange of good information. Unfortunately, no satisfactory theory exists to explain the discrepancies between the Uniform Crossover and the traditional approaches.
In the uniform crossover scheme individual bits in the string are compared between two parents. The bits are swapped with a fixed probability, typically 0.5.

Other crossover variants:

- *Arithmetic crossover*
  The chosen arithmetic operation on two parents is performed to create a new offspring. Figure 2.15 illustrates one point crossover.

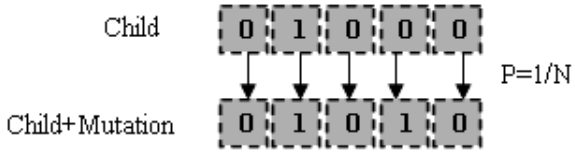

Figure 2.16: Mutation

Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to better solution by using mutation. Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. This reasoning also explains the fact that most GA systems avoid only taking the fittest of the population in generating the next but rather a random (or semi-random) selection with a weighting toward those that are fitter.

- *Flip bit mutation*
  Each gene in chromosome of the individual with the certain probability is inverted (for binary chromosome representation). The probability is usually proportional to the chromosome's length.

- *Simple bit mutation*
  This operator works similar as a flip bit mutation. It is also designed for a binary chromosome representation. The difference is in the probability of the bit inversion which in this case equals to $\frac{1}{2}$.

There exist different mutation operators described in literature:

- *Swapping mutation*
  Swapping mutation can be performed on different types of chromosome representation. Two random positions on chromosome are selected and genes on these positions are swapped.

- *Sliding mutation*
  Sliding mutation can also be applied with many types of chromosome representation. It is similar to the swapping mutation, two random points are selected: i and j, $i < j - 1$. The genes between $i + 1$ to j slided to the left by one position, and gene$i$ goes to the position j. It can be formulated as:

$$c_k = c_{k-1} \text{ if } k = \{i + 1, \ldots, j\} \text{and } c_j = c_i, \tag{2.80}$$

  where c_k is gene on position k in the chromosome.

- *Scramble bit mutation*
  This operator selects a segment of genes and combines them randomly and the rest of the chromosome remains unchanged.

2.4.2 Co-Operation of Biology Related Algorithms

Metaheuristic algorithms start to demonstrate their power in dealing with complex optimization problems. There are a lot of different heuristic methods designed for unconstrained real-valued optimization including Particle Swarm Optimization (PSO) Kennedy *et al.* (1995), Wolf Pack Search (WPS) Yang *et al.* (2007), Firefly Algorithm (FFA) Yang (2009), Cuckoo Search Algorithm (CSA) Yang & Deb (2009) and Bat Algorithm (BA) Yang (2010), which imitate a nature process or behaviour of an animal group. Numerical experiments conducted in Akhmedova & Semenkin (2013) have shown that for a given task the best heuristic cannot be chosen in advance. Thus a metaheuristic that combines major advantages of PSO, WPS, FFA, CSA and BA provides better performamce than the individual algorithms. The method proposed in Akhmedova & Semenkin (2013) is called Co-Operation of Biology Related Algorithms (COBRA) Akhmedova & Semenkin (2013) and it is based on generating five populations (one population for each algorithm) that perform in parallel cooperating with each other (so-called island model). Figure 2.17 illustrates the main idea of COBRA.

First, short descriptions of the individual algorithms are provided. Original version of COBRA, binary modification and constrained modification that are used for SVM- and ANN-based classifier generation are described. Finally, automated design of ANN and SVM applied to the text classification task is discussed in detail.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an optimization method, where the population of candidate solutions (particles) moves in the feature space in the way that each particle iteratively moves according to the simple formula of the particle's position and velocity. The movement

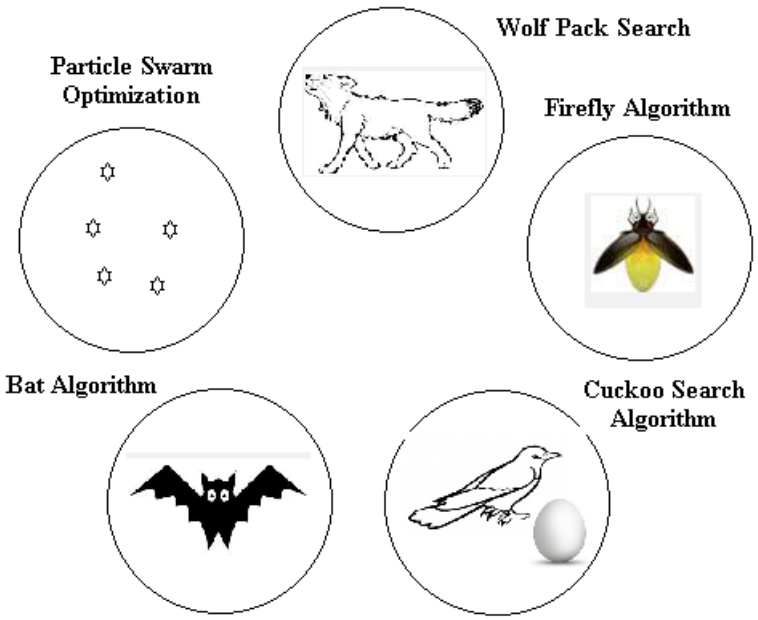

Figure 2.17: Co-Operation of Biology Related Algorithms (COBRA)

of each particle is affected by its local best known position and the best known position obtained by all particles. The idea of this method has been proposed by Kennedy *et al.* (1995) and assumes that the swam will move to the best solutions.

Wolf Pack Search

Wolf Pack Search is a local search method, which has been proposed by Yang *et al.* (2007) and has been designed to improve the local convergence of the Marriage in Honey Bees Optimization algorithm.

WPS-MBO uses three operators: Crossover, Mutation and Heuristic. Crossover and Mutation are the same as in GA. The Heuristic operator performs the local search.

Firefly Algorithm

Firefly algorithm is a metaheuristic optimization algorithm, which also has the population of candidate solutions that are able depending on their value of objective function (brightness) attract or be attracted to other candidate solutions. There are many variants of this algorithm. It has been proposed by Yang (2009).

Cuckoo Search Algorithm

Cuckoo search is an optimization algorithm proposed by Yang & Deb (2009). It is based on the heuristic that the egg represents a candidate solution and cuckoo egg is considered as a new solution presumably better. The probability that the cuckoo egg will not be discovered by the host bird, i.e. the new candidate solution will replace one of the existing ones, is a parameter of this algorithm. The set of existing solutions that can be replaced by the cuckoo egg is formed with the individuals with the worst value of the objective function.

Bat Algorithm

Bat algorithm is an optimization algorithm developed by Yang (2010). It has a population of candidate solutions, each of them has velocity, position and a varying frequency and loudness. They move randomly and change their frequency, emission rate and loudness. BA uses a frequency-tuning technique to control the dynamic behaviour of a swarm of bats and find a balance between local and global convergence.

2.4.2.1 Original Co-Operation of Biology Related Algorithms

A metaheuristic of Co-Operation of Biology Related Algorithms (COBRA) has been developed on the base of five well-known optimization methods including Particle Swarm Optimization (PSO) Kennedy *et al.* (1995), Wolf Pack Search (WPS) Yang *et al.* (2007), Firefly Algorithm (FFA) Yang (2009), Cuckoo Search Algorithm (CSA) Yang & Deb (2009) and Bat Algorithm (BA) Yang (2010). These algorithms are biology related optimization approaches originally designed for real-valued vector space. They imitate collective behaviour of corresponding animal groups, which allows these heuristic methods to find the global optimum of the objective function. Since these algorithms are similar to each other and according to Akhmedova & Semenkin (2013) the best algorithm cannot be chosen in advance for a given task, the metaheuristic approach uses cooperation that combines major benefits of PSO, WPS, FFA, CSA and BA.

This metaheuristic optimization algorithm is described as follows. COBRA generates five populations, where each population corresponds to each individual algorithm. These populations evolve in parallel cooperating with each other. It is a self-tuning method, i.e. the population sizes are calculated according to the value of the optimization criterion. The number of individuals in each algorithm's population increases or decreases as the fitness values increase or decrease. If the fitness function has not improved for the given number of generations, then the size of all populations will increase. If the fitness function has constantly improved, then the size of all populations will decrease. Besides, each population can "grow" by accepting individuals removed from other population. Population "grows" only if its av-

erage fitness value is better than the average fitness value of all other populations. Thereby the "winner algorithm" is determined on each iteration/generation.

The result of this kind of competition allows presenting the biggest resource (population size) to the most appropriate (in the current generation) algorithm. This property can be very useful in case of the complex optimization problem, where there is no single best algorithm on all stages of the optimization process.

One of the most important driving forces of this metaheuristic is the migration operator that creates a cooperation environment for component algorithms. All populations communicate with each other: they exchange individuals in such a way that some part of the worst individuals of each population is replaced by the best individuals of other populations. It updates information about the best achievements to all component algorithms and prevents their preliminary convergence to their own local optima, which improves the group performance of all algorithms.

2.4.2.2 Constrained modification of Co-Operation of Biology Related Algorithms

Original version of COBRA has been designed for unconstrained optimization. However, many applications involve constrained optimization problems such as finding the optimal parameters of the given kernel function in the task of automated SVM-based classifier design.

The modification of COBRA for solving constrained real-parameter optimization problems has been developed. For this purpose three constraint handling techniques have been used:

- dynamic penalties proposed in Eiben & Smith (2003),
- Deb's rule introduced by Deb (2000),
- technique described in Liang *et al.* (2010).

The method proposed in Liang *et al.* (2010) has been implemented in the PSO-component of COBRA. WPS, FFA, CSA and BA component algorithms have been modified by implementing firstly Deb's rule and then calculating function values by using dynamic penalties.

The performance of unconstrained and constrained version of COBRA has been evaluated on a set of benchmark problems from CEC'2013 and CEC'2009 competitions. Numerical experiments conducted in Akhmedova & Semenkin (2013) have shown that COBRA and its constrained modification outperform all component algorithms.

2.4.2.3 Binary Modification of Co-Operation of Biology Related Algorithms

Since all above listed heuristic algorithms (PSO, WPS, FFA, CSA and BA) have been originally designed for real-valued spaces. However, a lot of real world applications are defined

in discrete valued spaces, where the domain of the variables is finite. In order to solve such problems a binary modification of COBRA has been developed.

Original COBRA has been adapted to search in binary spaces by applying a sigmoid transformation to the velocity component (PSO, BA) and coordinates (FFA, CSA, WPS) to squash them into a range [0, 1] and force the component values of the positions of the particles to be 0's or 1's.

The basic idea of this adaptation has been taken from Kennedy & Eberhart (1997), where it has been used for PSO algorithm. In PSO method each particle has a velocity as it has been defined in Kennedy *et al.* (1995) and binarization of individuals is conducted by calculating the value of sigmoid function as in Kennedy & Eberhart (1997). The applied sigmoid function is given by

$$S(v) = \frac{1}{1 + \exp(-v)}, \tag{2.81}$$

where v is a velocity of the individual.

Then, a random number r from the range [0, 1] is generated and the corresponding component value of particle's position is 1 if r is smaller than $S(v)$ and 0 otherwise. The particle's position is calculated as

$$p(v, r) = \begin{cases} 1, & \text{if } r < S(v) \\ 0, & \text{otherwise} \end{cases}. \tag{2.82}$$

Since in BA each bat also has a velocity Yang (2010), exactly the same procedure can be directly applied for the binarization of this algorithm. For WPS, FFA and CSA Yang *et al.* (2007); Yang (2009); Yang & Deb (2009) algorithms the sigmoid transformation is applied in the same way to position components of individuals and then a random number r is compared with obtained value.

2.5 Summary

In this chapter the state of the art in the field of text classification including text preprocessing techniques, supervised classification algorithms, semi-supervised and unsupervised classification methods was reviewed.

In general, text preprocessing involves feature extraction, term weighting and dimensionality reduction of the feature space, which are presented and discussed in the first part of the chapter. In this thesis the main focus is on a novel term weighting approach and a feature space dimensionality reduction method. The term weighting technique introduced in this work belongs to the field of supervised term weighting, i.e. it uses information about class

annotations of the training data. The dimensionality reduction algorithm proposed in this thesis is based on term clustering, therefore, it combines supervised and unsupervised learning and falls into dimensionality reduction algorithms by feature extraction.

The second part of the chapter presents several well known classification algorithms designed for supervised text classification.

The ways to evaluate the algorithms performance are shown. In order to evaluate any text preprocessing method it should be applied using different classification algorithms, since the text preprocessing techniques and classification algorithms depend on each other, i.e. one classifier performs better with one preprocessing technique and another classification algorithm works better with the different type of preprocessing. The common set of the classification methods used to evaluate the text preprocessing method consists of Support Vector Machine (SVM), Artificial Neural Network (ANN), k-Nearest Neighbor algorithm, Rocchio classifier and Bayes classifier. Another direction of this thesis concerns the automatic design and the improvement of classification quality of existing classifiers such as SVM and ANN since they provide the best results on different task with various preprocessing techniques.

In the third part of the chapter, some popular algorithms in the areas of unsupervised and semi-supervised machine learning have been discussed. First, the different approaches to measure the performance of clustering algorithms have been shown. An overview of the different clustering strategies has been presented (connectivity-based, centroid-based, distribution-based, density-based and other clustering algorithms such as graph-based clustering methods).

Different semi-supervised approaches have been described (Self-training, Co-training and Generative models). The difference of these approaches is in their definitions of the objective functions. However, as described above, the objective is formulated in all cases as a global function which takes into consideration labelled and unlabelled data simultaneously. In the methods developed in this thesis the first stage is to create the text representation, where the class annotations are used in the proposed formula. The next step is to separate terms that "point" to different classes and group terms that "point" to one class. Then, the hierarchical clustering for each group of terms is applied. After that, the class labels are used during the optimization of the obtained term clusters weights. For some of the databases it is necessary to add an unsupervised criterion to the objective function.

Finally, this chapter presents a brief overview of heuristic optimization algorithms that are used for automated generation and parameters selection of text classifiers. These heuristic algorithms include genetic algorithm (GA) and other biology related methods. A metaheuristic method called Co-Operation of Biology Related Algorithms (COBRA) and its modifications have also been described. This metaheuristic approach combines five different heuristic algorithms: Particle Swarm Optimization (PSO) proposed in Kennedy *et al.* (1995), Wolf Pack Search (WPS) proposed in Yang *et al.* (2007), Firefly Algorithm (FFA) proposed in Yang

(2009), Cuckoo Search Algorithm (CSA) proposed in Yang & Deb (2009) and Bat Algorithm (BA) proposed in Yang (2010). The original version of COBRA has been designed for unconstrained optimization in real-valued vector space, a constrained version uses Deb's rule, dynamic penalties (both for WPS, FFA, CSA and BA) and the technique proposed in Liang *et al.* (2010) (for PSO) to modify the component algorithms. For a binary modification the value of sigmoid function from the position of particles is calculated and this value is considered as a probability that the corresponding position equals to 0. In this thesis the Artificial Neural Network and the Support Vector Machine classifiers generated using the COBRA versions are modified for the text classification task.

Currently, text categorization research is going in several directions. The effort to improve existing classifiers is one of the directions. Researchers attempt to improve the performance, find the best parameters for different types of problems and increase their speed in training or testing phase.

Another direction is a search of better text representation and better preprocessing techniques. Generally, there exist several methods to evaluate the importance of a particular term with respect to the text categorization task.

Finally, researchers try to combine supervised and unsupervised learning for different purposes. Some try to include unlabelled data to the learning process, others apply clustering algorithms to reduce the feature space dimensionality or to find better result with the supervised classifier.

This thesis is concerned with the text representation problem, improvement of the existing classifiers (Support Vector Machine and Artificial Neural Network) and dimensionality reduction with help of unsupervised learning models. The aim of this work is to decrease the human supervision in the text preprocessing, including the dimensionality reduction techniques and the text classification methods. A lot of modern methods for text preprocessing require manually designed filtering (specially formed for each domain and language stop word and ignore word lists) and stemming algorithms which are specific for every language. Despite of the fact that the well selected classification algorithm with the carefully chosen text preprocessing technique is able to produce excellent performance on the given task, since the choice of the classifier and text representation have been based on it. However, the problem of this approach is the difficulty to directly apply the method to another problem without the loss in the classification performance. This thesis focuses on the automatic statistical text preprocessing and dimensionality reduction as well as an automatic generation and parameters adjustment of the popular classifiers (SVM and ANN), therefore, the developed methods can be transported to another domain or language without major changes.

In order to evaluate the developed text preprocessing and text classification algorithms the several corpora have been selected. They are designed for different tasks and are written in

different languages, they consist of grammatically correct sentences and of transcribed natural language utterances. The corpora are described in detail in the following chapter.

Chapter 3

Corpora Description

In this thesis, different real-world data sets are used to evaluate the approaches described in the following chapters. These corpora are related to the different text classification problems: natural language call routing, opinion mining and topic categorization. These data sets are outlined in this chapter.

In order to test the performance of the text preprocessing and text classification algorithms, corpora should be as different as possible. In this work for algorithms application and comparison of experimental results DEFT ("Défi Fouille de Texte") Evaluation Package 2008 (def (2008)) and publically available corpora from DEFT'07 (def (2007)) have been used. The DEFT Evaluation Package 2008 has been provided by ELRA (European Language Resources Association). SpeechCycle company (was acquired by Synchronoss in 2012) provided transcripts of client calls obtained from a trouble shooting agent. Table 3.1 presents the features of the corpora used in this thesis.

These corpora are related to the text classification field, however, they are designed for different tasks: opinion mining, topic categorization and call routing. Each of these tasks has its own specialities. Topic categorization corpora (DEFT 2008) have large term set (~200000-250000 terms) and each document contains many terms, i.e. a lot of statistical information and since the feature space dimensionality is so high, the majority of classification algorithms

Corpus	**Source**	**Task**	**Language**	**Number of classes**
Books	DEFT 2007	Opinion mining	French	3
Games	DEFT 2007	Opinion mining	French	3
Debates	DEFT 2007	Opinion mining	French	2
T1	DEFT 2008	Topic categorization	French	4
T2	DEFT 2008	Topic categorization	French	5
SpeechCycle	SpeechCycle	Call Routing	English	20

Table 3.1: Common table of the corpora

have problems to process such vectors. Opinion mining corpora (DEFT 2007) have also large enough term set (~50000-60000 terms) with long documents. Most of text classification algorithms, which are designed for opinion mining, for each term try to catch its emotional direction. However, general text classification methods are also used in opinion mining tasks. Topic categorization and opinion mining corpora are written in French language and they have a small number of categories (~2-5 classes). Call routing corpus has a relatively small feature space (~3500 terms) and most utterances are short (~5 terms), since callers know that they speak with an automatic trouble shooting agent many utterances include just key words. Documents of this corpus are transcribed from real speech (natural language) in contrast with topic categorization and opinion mining data sets with grammatically correct sentences. This corpus is written in English language and has relatively many categories (20 classes).

3.1 Opinion Mining

Opinion mining which is also called sentiment analysis related to the use of natural language processing, text analysis and computational linguistics to find and extract subjective information in the given data.

Generally, the aim of opinion mining is to determine the attitude of a speaker or a writer with respect to some topic or the overall context of a document. The attitude may be his or her judgment or evaluation, affective state (the emotional state of the author when writing), or the intended emotional communication (the emotional effect the author wishes to have on the reader).

The accuracy of an opinion mining system is, in general, how well it agrees with human judgments. This is usually measured by precision and recall. However, according to research human raters typically agree 79% of the time. Therefore, a 70% accurate program is performing nearly as well as humans, even though such accuracy rate does not look impressive. Suppose even that the program is "right" 100% of the time, humans would still disagree with it about 20% of the time, since they disagree that much about any answer (even between themselves). More sophisticated measures or scores can be used to evaluate the performance of the opinion mining system, however it remains a complex problem.

The rise of social media such as blogs and social networks has attracted interest in opinion mining systems. With the reviews, ratings, recommendations and other forms which are available online, it has become very important for companies to have methods to process this kind of information to improve their business strategies, market their products, identify new opportunities and manage their reputations. As companies look to automate the process of filtering out the noise, understanding the conversations and identifying the relevant content, many researchers are now looking to the field of sentiment analysis.

Books	**Training set**	**Test set**	%
Positive	1150	766	55%
Neutral	615	410	30%
Negative	309	206	15%

Table 3.2: Proportion of the articles per category: Books DEFT 2007

Games	**Training set**	**Test set**	%
Positive	874	582	34%
Neutral	1166	777	46%
Negative	497	331	20%

Table 3.3: Proportion of the articles per category: Games DEFT 2007

The problem is that most opinion mining algorithms use simple terms to express sentiment about a product or service (for example, "positive" or "negative" review). However, cultural factors, linguistic nuances and differing contexts make it extremely difficult to turn a string of written text into a simple pro or con sentiment. The fact that humans often disagree on the opinion expressed in the current text shows the complexity of the task and it is extremely difficult for computers to perform with the good accuracy. The shorter the text string, the harder it becomes.

The focus of DEFT 2007 campaign is the sentiment analysis, also called opinion mining. In this thesis three publically available corpora are used:

- reviews on books and movies (in this thesis this corpus is referred as Books),
- reviews on video games (Games),
- political debates about energy project (Debates).

All databases are divided into the training (60% of the whole number of articles) and the test set (40%) (this division into training and test sets has been made by organizers of DEFT 2007 campaign and in this thesis it has been kept this way in order to directly compare algorithm performances obtained by the participants with the developed methods).

The training and test sets of all corpora are shown in Tables 3.2-3.4.

In order to apply the classification algorithms all words which appear in the training set have been extracted. Then words have been brought to the same letter case; dots, commas

Debates	**Training set**	**Test set**	%
For	6899	4595	40%
Against	10400	6926	60%

Table 3.4: Proportion of the articles per category: Debates DEFT 2007

Corpus	Size	Classes
Books	Train size = 2074 Test size = 1386 Term set size = 52507	0: negative 1: neutral 2: positive
Games	Train size = 2537 Test size = 1694 Term set size = 63144	0: negative 1: neutral 2: positive
Debates	Train size = 17299 Test size = 11533 Term set size = 59615	0: against 1: for

Table 3.5: Corpora description (DEFT'07)

and other punctual signs have been removed. It should be mentioned that no other information related to the language or domain (no stop or ignore word lists) has been used in the preprocessing

The description of the corpora is presented in Table 3.5.

In the Table 3.6 results (strict F-scores) obtained by participants of the DEFT 2007 Campaign published in def (2007) are shown. Each of the participants could have submitted maximum three variants of their systems. The best F-scores for each data set are written in bold. Average performance is shown in italics on the last row.

Table 3.6 shows that the best F-scores on all corpora have been obtained by Torres-Moreno team (Books: $Fscore = 0.603$; Games: $Fscore = 0.784$; Debates: $Fscore = 0.720$).

Performance achieved by participants varies:

1. Corpus Books : from 0.377 to 0,603;
2. Corpus Games : from 0.457 to 0,784;
3. Corpus Debates: from 0.540 to 0,784.

Human experts have also performed these tasks in order to estimate the difficulty:

1. Corpus Books : the strict F-scores obtained by humans vary from 0.52 to 0,79;
2. Corpus Games : the strict F-scores obtained by humans vary from 0.73 to 0,90;
3. Corpus Debates: there is no data concerning human performance.

Here one can see that Books corpus appeared to be a difficult classification task for both human and algorithms (the average result obtained by participants is only 0.02 lower than the worst human performance); the human experts perform much better on Games corpus where the average result of participants reached only 0.66 which is 0.07 lower than the lowest result of human performance.

Team	Submission	Books	Games	Debates
J.-M. Torres-Moreno (LIA)	1	0.602	**0.784**	0.719
J.-M. Torres-Moreno (LIA)	2	**0.603**	0.782	**0.720**
J.-M. Torres-Moreno (LIA)	3	**0.603**	0.743	0.709
G. Denhière (EPHE and U. Würzburg)	1	0.476	0.640	0.577
G. Denhière (EPHE and U. Würzburg)	2	0.599	0.699	0.681
S. Maurel (CELI France)	1	0.513	0.706	0.697
S. Maurel (CELI France)	2	0.418	0.538	0.697
S. Maurel (CELI France)	3	0.519	0.700	0.697
M. Vernier (GREYC)	1	0.577	0.761	0.673
M. Vernier (GREYC)	2	0.532	0.715	0.639
M. Vernier (GREYC)	3	0.532	0.715	0.673
E. Crestan (Yahoo ! Inc.)	1	0.529	0.670	0.652
E. Crestan (Yahoo ! Inc.)	2	0.523	0.673	0.703
M. Plantié (LGI2P and LIRMM)	1	0.421	0.783	0.618
M. Plantié (LGI2P and LIRMM)	2	0.424	0.732	0.671
M. Plantié (LGI2P and LIRMM)	3	0.472	0.547	0.608
A.-P. Trinh (LIP6)	1	0.542	0.659	0.676
A.-P. Trinh (LIP6)	2	0.490	0.580	0.665
M. Généreux (NLTG)	1	0.453	0.623	0.540
M. Généreux (NLTG)	2	0.464	0.626	0.554
M. Généreux (NLTG)	3	0.441	0.602	0.569
E. Charton (LIA)	1	0.377	0.619	0.616
E. Charton (LIA)	2	0.504	0.457	0.553
E. Charton (LIA)	3	0.504	0.619	0.553
A. Acosta (Lattice)	1	0.392	0.536	0.582
Average result		*0.5004*	*0.6604*	*0.6417*

Table 3.6: Strict F-scores ($\beta = 1$) obtained by participants of DEFT'07

It should be mentioned that the most difficult category for classification algorithms to detect is a "neutral" one since the neutral reviews consist of both positive and negative terms which makes it extremely difficult even for humans to objectively choose the class of a document.

3.2 Topic Categorization

Nowadays there are many applications that solve different topic categorization tasks. In this thesis the developed algorithms have been applied to the corpora for document classification and natural language call routing.

Generally, topic categorization task is considered as the set of textual units which are being classified into the list of pre-existing topics (also called categories, themes, classes).

T1	Training set	%	Test set	%
Art	5767	38%	3844	36%
Economy	4630	30%	3085	29%
Sport	3474	23%	2315	22%
Television	1352	9%	1352	13%

Table 3.7: Proportion of the articles per category: T1 DEFT 2008

3.2.1 Document Classification

The topic of DEFT 2008 edition is related to the text classification by categories and genres. The data consists of two corpora (in this thesis the first task is referred as T1 and the second task as T2) containing articles of two genres: articles extracted from French daily newspaper Le Monde and encyclopaedic articles from Wikipedia in French language (T1 and T2 both include articles of mixed genres). This thesis reports on the results obtained using both tasks of the campaign and focuses on detecting the category.

The categories in which the articles of the corpora are divided into:

- *Art*: articles devoted to the art and culture (dance, painting, sculpture, theatre);
- *Economy*: articles on the economy and business;
- *France*: French articles national policy;
- *International*: articles international or national policy (except French policy);
- *Literature*: articles on books (reviews, publications) and literature;
- *Science*: articles on science;
- *Social*: articles on social issues not covered by the policy area;
- *Sports*: articles about sports (events, results, figures);
- *Television*: articles on radio and television (program operation).

All databases are divided into the training (60% of the whole number of articles) and the test set (40%). (this division into training and test sets has been made by organizers of DEFT 2008 campaign and in this thesis it has been kept this way in order to compare algorithm performances obtained by the participants with the developed methods).

The training and test sets of both corpora are shown in Tables 3.7 and 3.8.

In order to apply the classification algorithms all words which appear in the training set have been extracted. Then words have been brought to the same letter case; dots, commas

T2	Training set	%	Test set	%
France	3326	14%	2216	14%
International	5305	23%	3536	23%
Literature	4576	19%	3049	19%
Science	6565	28%	4375	28%
Society	3778	16%	2517	16%

Table 3.8: Proportion of the articles per category: T2 DEFT 2008

Corpus	Size	Classes
T1	Train size = 15223 Test size = 10596 Term set size = 202979	0: Sport 1: Economy 2: Art 3: Television
T2	Train size = 23550 Test size = 15693 Term set size = 262400	0: France 1:International 2: Literature 3: Science 4: Society

Table 3.9: Corpora description (DEFT'08)

and other punctual signs have been removed. It should be mentioned that no other information related to the language or domain (no stop or ignore word lists) has been used in the preprocessing

The description of the corpora is presented in Table 3.9.

In the Table 3.10 results (strict F-scores) obtained by participants of the DEFT 2008 Campaign published in def (2008) are shown. Each of the participants could have submitted maximum three variants of their systems. The best F-scores for each data set are written in bold. Average performance is shown in italics on the last row.

Table 3.10 shows that the best F-scores on the corpus T1 has been obtained by D. Buffoni and on corpus T2 by Torres-Moreno team (T1: $Fscore = 0.894$; T2: $Fscore = 0.880$).

Performance achieved by participants varies:

1. Corpus T1 : from 0.672 to 0,894;
2. Corpus T2 : from 0.328 to 0,880.

Human experts have also performed these tasks in order to estimate the difficulty:

1. Corpus T1: the strict F-scores obtained by humans vary from 0.66 to 0,82;
2. Corpus T2: the strict F-scores obtained by humans vary from 0.84 to 0,89.

Team	Submission	T1	T2
J. M. Torres-Moreno (LIA)	1	0.859	0.859
J. M. Torres-Moreno (LIA)	2	0.883	0.872
J. M. Torres-Moreno (LIA)	3	0.854	**0.880**
M. Plantié (LGI2P/LIRMM)	1	0.853	0.858
M. Plantié (LGI2P/LIRMM)	2	0.852	0.852
M. Plantié (LGI2P/LIRMM)	3	0.823	0.828
E. Charton (LIA)	1	0.875	0.879
E. Charton (LIA)	2	0.809	0.662
E. Charton (LIA)	3	0.844	0.853
D. Buffoni (LIP6)	1	0.804	0.874
D. Buffoni (LIP6)	2	0.879	0.874
D. Buffoni (LIP6)	3	**0.894**	0.876
G. Cleuziou (LIFO/INaLCO)	1	0.790	0.821
F. Rioult (GREYC)	1	0.849	0.838
F. Rioult (GREYC)	2	0.672	0.328
F. Rioult (GREYC)	3	0.672	0.815
Average result		*0.8258*	*0.8106*

Table 3.10: Strict F-scores ($\beta = 1$) obtained by participants of DEFT'08

It is shown that for humans the second task is easier than the first one, however, for automatic systems the situation is different: on first task the performance of algorithms developed by participants was better than even the human performance; but second task has appeared to be more complicated for an automatic classification.

3.2.2 Natural Language Call Routing

The data set for Natural Language Call Routing is a corpus of transcribed calls collected from spoken dialogue system. Spoken Dialogue Systems became a new important form of communication between human and machine. The typical architecture of an SDS Minker *et al.* (2006) is presented in Figure 3.1.

First, input acoustic vectors generated from the speech signal are processed by an Automatic Speech Recogniser (ASR), which obtains a raw text transcription of the input call. Next, the transcribed text comes to a Natural Language Understanding block (semantic analysis) which extracts the meaning of the call in form of an appropriate semantic structure (in our case semantic structure is a list of problem categories). Then this semantic representation is processed by the dialog manager which also communicates directly with an external application, namely a database interface. The dialog manager controls the overall progress of interaction according to the task completion. During this process, the user may be asked for confirmations, disambiguations, additional information, etc. Finally, the result of interaction

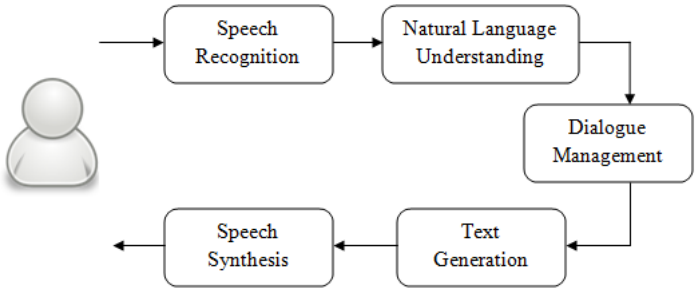

Figure 3.1: Overview of Spoken Dialogue Systems

Transcribed call	Category
I need to get a hold of somebody to uh fix my...	Appointments
I need to find a payment center	Bill
Corporate office phone num...	_TE_NOMATCH
Uh...	_TE_NOMATCH
Technical support	TechSupport
Agent... need to speak with an agent	Operator
Make a change to my service	ChangeService

Table 3.11: Some examples of calls and corresponding categories manually chosen by experts

is given to the user in form of speech by text-to-speech synthesis or pre-recorded prompts.

The call corpus used in this thesis has been obtained from a special type of Spoken Dialogue Systems which is called an automated troubleshooting agent. These are SDS specially designed to perform customer support issues over the telephone, in a similar way as human operators do. The most important part of the automated troubleshooting agent is the natural language understanding block. Typically, the semantic analysis in this kind of systems is carried out at the level of calls. The callers are presented an open prompt, such as "please briefly described the reason for your call". Then, the natural language understanding block matches the unconstrained, natural language user response into one of the possible problems from a predefined list. This is done by means of statistical classifiers. This particular kind of semantic analysis is generally known as Statistical Spoken Language Understanding.

The machine learning algorithms developed in this thesis are focused on the supervised preprocessing and the combination of unsupervised and supervised preprocessing of calls. The given corpus has been made available by SpeechCycle Company and it is related to the application domain named "Call Router Backup Menu" of commercial troubleshooting agent. Some examples of transcribed calls and their corresponding categories manually chosen by experts are presented in Table 3.11.

It should be noted that although these calls transcriptions may have a similar structure as text sentences, they share the characteristics of natural language, in contrast to sentences in

written language. Therefore, it is common to observe bad-formed or grammatically incomplete call transcriptions. Another difference is a length of a typical call in comparison with the usual document: clients usually know that they are talking to the automatic system and sometimes the call is more a set of keywords rather than a grammatical sentence. Moreover, generally calls consist of only one sentence, while documents usually have many sentences. A lot of calls contain only one or two words.

Natural language call routing can be treated as an instance of topic categorization of documents (where the collection of labelled documents is used for training and the problem is to classify the remaining set of unlabelled test documents) but it also has some differences. For instance, in document classification there are usually much more terms in one object than in single utterance from call routing task, where even one-word utterances are common. Therefore, for call routing words appeared in the utterance are generally more important to the final decision.

The database for training and performance evaluation consists of about 300.000 user utterances recorded from caller interactions with commercial automated agents. Calls are represented as a text after speech recognition. The utterances have been automatically transcribed and manually labelled by experts into 20 classes (call reasons), such as appointments, operator, bill, internet, phone or video. Calls that cannot be routed certainly to one reason of the list are classified to class _TE_NOMATCH.

Table 3.12 shows some features of the SpeechCycle corpus by categories.

Class TE_NOMATCH has some special features:

- It contains non-informative calls
- It includes 45% words, which do not appear in other classes
- Most calls from the databases are assigned to it: 27% in the whole database
- The classification algorithms have difficulties in detecting it (it has the biggest classification error) due to its non-uniform structure:
  - Some utterances have no meaning
  - Some utterances cannot be routed to any of other classes because there is no special class for this type of problem
  - The rest of utterances have key words of more than one meaningful class and cannot be assigned to only one class

Table 3.13 shows expectation and variance of some features in class TE_NOMATCH and other 19 classes. Word is considered frequent if its relative frequency R_j (the ratio between

Call reasons	Number of terms (training set)	Number of terms (test set)	Number of unknown terms
_TE_NOMATCH	2337 (67%)	711 (21%)	139 (4%)
serviceVague	274 (8%)	95 (3%)	29 (0.8%)
appointments	322 (9%)	97 (3%)	14 (0.4%)
none	250 (7%)	73 (2%)	18 (0.5%)
cancelService	177 (5%)	54 (2%)	10 (0.3%)
idk	163 (5%)	41 (1%)	5 (0.1%)
orders	364 (11%)	134 (4%)	18 (0.5%)
UpForDiscussion _Complaint	56 (2%)	15 (0.4%)	1 (0.03%)
operator	721 (21%)	233 (7%)	29 (0.8%)
techSupport	667 (19%)	199 (6%)	27 (0.8%)
bill	759 (22%)	246 (7%)	36 (1%)
internet	289 (8%)	88 (3%)	18 (0.5%)
phone	317 (9%)	85 (2%)	20 (0.6%)
techSupport_internet	248 (7%)	75 (2%)	17 (0.5%)
techSupport_phone	290 (8%)	94 (3%)	16 (0.5%)
techSupport_video	587 (17%)	214 (6%)	27 (0.8%)
video	506 (15%)	152 (4%)	38 (1%)
changeService	279 (8%)	81 (2%)	10 (0.3%)
UpForDiscussion _no_audio	449 (13%)	109 (3%)	19 (0.5%)
UpForDiscussion _AskedToCall	103 (3%)	25 (0.7%)	1 (0.02%)

Table 3.12: Some features of the SpeechCycle corpus by categories

n_j, the number of documents which contain this word, and N, the number of all documents in the corpus) is greater than some constant (in this case word j is frequent if $R_j = \frac{n_j}{N} > 0.9$).

From the Table 3.13 one can see that the mathematical expectation of the number of frequent words is significantly different for the class TE_NOMATCH ($EV = 2,645$) and other categories ($EV = 4,263$). Since generally the decision of the call reason of the input utterance is made based on the words which usually have usually occurred in the utterances of this class, the TE_NOMATCH class is more difficult to detect. The number of unknown words (words which did not appear in the training set) is also much higher in TE_NOMATCH category, which makes the classification process more complicated.

On the preprocessing stage all utterance duplicates have been removed. The preprocessed database consisting of 24458 utterances was divided into the training (22020 utterances, 90,032%) and the test set (2438 utterances, 9,968%) such that the percentage of classes remains the same in both sets.

The size of the vocabulary of the whole database (the term set) is 3464 words:

	% of unknown words	**Number of frequent words**
EV of class TE_NOMATCH	9,9	2,645
EV of other classes	1,8	4,263
Variance of class TE_NOMATCH	6,1	4,450
Variance of other classes	0,8	5,490

Table 3.13: Some features of the SpeechCycle corpus

- 3294 words appear in the training set,
- 1124 words appear in the test set,
- 170 words appear only in the test set and do not appear in the training set (unknown words),
- 33 utterances consisted of only unknown words (and therefore cannot be correctly classified), and 160 utterances included at least one unknown word.

The database is also unbalanced, some classes include much more utterances than others (the largest class _TE_NOMATCH includes 27.85% utterances and the smallest one consists of only 0.16% utterances).

In the next chapter novel text preprocessing methods including a novel term weighting and novel dimensionality reduction algorithms as well as novel text classification methods that are used to process these corpora are introduced.

Chapter 4

Novel Text Preprocessing and Classification Methods

4.1 Novel Text Preprocessing Methods

4.1.1 Introduction

The task of text classification can be divided into the text preprocessing part and classification itself. Figure 4.1 presents the scheme of the text classification process. First, a text preprocessing algorithm is applied to the given textual data to produce a document representation in some feature space. Then, a machine learning algorithm is used to classify the preprocessed data.

Methods of text preprocessing influence on the performance of classification algorithms. Generally, text documents are processed in the way that they can be considered as vectors from some feature space. The construction of this feature space usually differs between text preprocessing algorithms. This way of document representation is called vector space model.

In the vector space model, documents are represented as a vector where each component is associated with the certain word (term) from the vocabulary obtained on the training set. Generally, each vector component (each term) is assigned a value related to the estimated importance (confidence or weight) of this word in the document. Traditionally, this weight was calculated using the TF-IDF (IDF is the weight of the certain term). TF-IDF is unsupervised

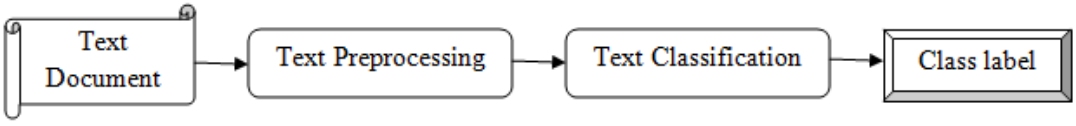

Figure 4.1: Common diagramm of text preprocessing and text classification

algorithm, therefore, it does not use information about classes. The text preprocessing scheme is presented in Figure 4.2.

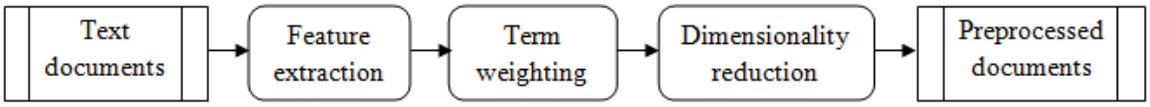

Figure 4.2: Text preprocessing diagramm

Let X be a text corpus and A and B are categorization tasks. Using the TF-IDF weighting each term will be represented the same way for both A and B tasks. Thus, according to TF-IDF the importance of the term is independent from the categorization task. However, this is mostly not the case. Suppose that X contained cooking recipes and scientific articles both written either in English or in German. Suppose that A is the task that consists of classifying X into cooking recipes and everything else where words as potatoes and tomatoes should have greater weight than words "the" or "of". Let B be the categorization task which consists of classifying X in two languages. Then words "the" and "der" should have greater importance. However, using TF-IDF for both tasks, words "the" and "der" would have very low weight due to their high occurrence in the corpus. The idea of our method as in for example ConfWeight is to benefit from the knowledge about the categorization task using the labels from the training set to assign term weights.

Figure 4.2 illustrates three main stages of text preprocessing. It involves feature extraction, term weighting and dimensionality reduction. In this work an initial feature extraction algorithm considers features as words and on a dimensionality reduction step clusters of words are denoted as features.

This chapter introduces a novel supervised term weighting method based on the estimation of the word importance for a certain problem. In order to calculate the weight of term the adaptation of the formula used for fuzzy rules relevance estimation and several decision rules based on this term weighting are discussed.

In this chapter a novel method for dimensionality reduction of the feature space is proposed. It involves term clustering using hierarchical agglomerative clustering based on the term weights obtained on the previous step. The second step consists of the optimization method applied to recalculate the weights of the obtained term clusters using coevolutionary genetic algorithm with a cooperative scheme.

This chapter also introduces a novel semi-supervised approach that combines supervised and unsupervised learning to improve the performance of classification methods. The designed algorithm aims to improve the detection of large non-uniform categories by clustering them into a set of well-defined subcategories and, then, classifying given documents into a set of other informative classes and an obtained set of subcategories.

In this thesis the natural language call routing task (SpeechCycle corpus) has one large category that consists of different types of documents (calls). In this class there are utterances that are irrelevant to the given call routing task, e.g. calls with no recognized words, calls with no meaning or no clear reason. The majority of classification algorithms have problems detecting this class.

In this chapter an approach to improve the performance of the classification algorithms is described. It divides this category into the set of subclasses, based on the assumption that the set of small well-defined classes is easier to detect than a large non-uniform category.

4.1.2 Novel Term Weighting (TW) Technique based on Fuzzy Rules Relevance

The main idea of the method is to take advantage from the knowledge about the given classification task, in other words - to use the class labels from the training set to find the term weights (supervised term weighting). The details of the approach are described in Gasanova *et al.* (2013c). The term weighting step in text preprocessing is presented in Figure 4.3.

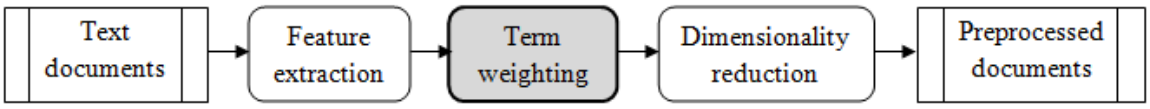

Figure 4.3: The term weighting step in text preprocessing

The approach is similar to ConfWeight but it is not so time-consuming. The idea is that every word that appears in the article has to contribute some value to the certain class and the class with the biggest value is defined as a winner for this article.

For each term a real number term relevance is assigned that depends on the frequency in utterances or documents. Term weight is calculated using a modified formula of fuzzy rules relevance estimation for fuzzy classifier Ishibuchi *et al.* (1999)). The membership function has been replaced by word frequency in the current class.

The details of the procedure are the following.

Let L be the number of classes, n_i is the number of articles which belong to the ith class and N_{ji} is the number of jth word occurrence in all articles from the ith class. The relative frequency of jth word occurrence in the ith class is defined as

$$T_{ji} = \frac{N_{ji}}{n_i}. \tag{4.1}$$

For each term the maximum of the relative frequencies is calculated $R_j = \max_i T_{ji}$ and to the jth term the number of class is assigned where this maximum is achieved $S_j =$

Name	**Decision rules**	
RC	$A_i = \sum_{j:S_j=i} R_j C_j$	For each class i A_i is calculated The number of class that achieves maximum of A_i is chosen $winner = \arg(\max A_i)$
RC max	$A_i = \sum_{j:S_j=i} \max R_j C_j$	
C	$A_i = \sum_{j:S_j=i} C_j$	
C with limit	$A_i = \sum_{j:S_j=i\, C_j>const} C_j$	
R	$A_i = \sum_{j:S_j=i} R_j$	

Table 4.1: Decision Rules

$\arg(\max_i T_{ji})$.

The term weight, C_j, is given by

$$C_j = \frac{1}{\sum_{i=1}^{L} T_{ji}}(R_j - \frac{1}{L-1}\sum_{i=1\, i\neq S_j}^{L} T_{ji}). \tag{4.2}$$

C_j is higher if the word occurs often in one class than if it appears in many classes.

The learning phase consists of counting the term weights for each term and calculating the document representation using the obtained term weights. This procedure means that this algorithm uses the statistical information obtained from the training set.

4.1.2.1 Decision Rule

This section shows several simple decision rules which can be used to find the class of the given document. For each category i the value A_i (based on the number of words which have been assigned to the category i and some value which represents the weight of these words) is calculated and the maximum of A_i, $i = \{1, \ldots, K\}$(K is the number of categories of the classification task) is found.

Table 4.1 shows the definition of the decision rules which have been tested.

The best obtained performance has been achieved with the decision rule C, where the class i of the document which has the highest sum of term weights C_j(terms belong to the document class i: $j : S_j = i$).

Figure 4.4 provides accuracies of different decision rules obtained on the SpeechCycle

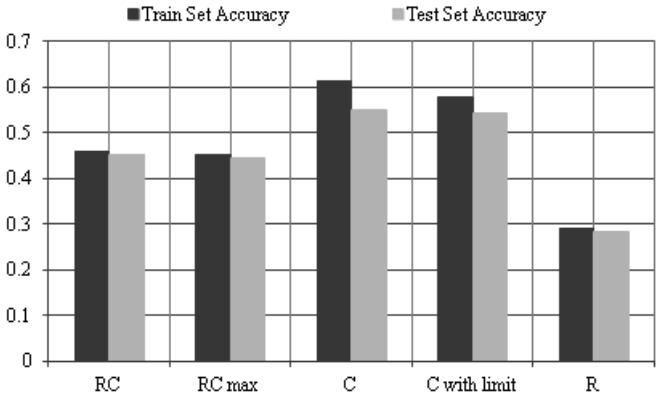

Figure 4.4: Comparison of decision rules (x-axis: decision rule; y-axis: accuracy) applied on SpeechCycle corpus

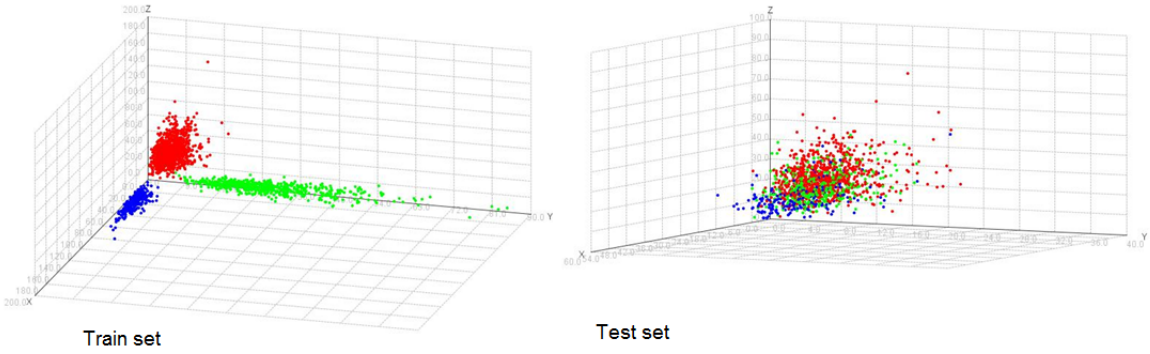

Figure 4.5: Document representation according to the novel TW: Books

corpus. Applying the best decision to the SpeechCycle database the train and test accuracies have reached 61% and 55% accordingly.

The best decision rule is given by

$$class\ winner = \arg\left(\max_i \sum_{j:S_j=i} C_j\right). \tag{4.3}$$

Figure 4.5 illustrates the document representation obtained on the Books corpus using the decision rule 4.3. One can see that despite of the well-sepated objects in the training set, the situation with test examples is not so good.

Figure 4.6 illustrates the document representation obtained on the Games corpus using the decision rule 4.3. The train elements are also separable according to this preprocessing, however, test objects are still mixed.

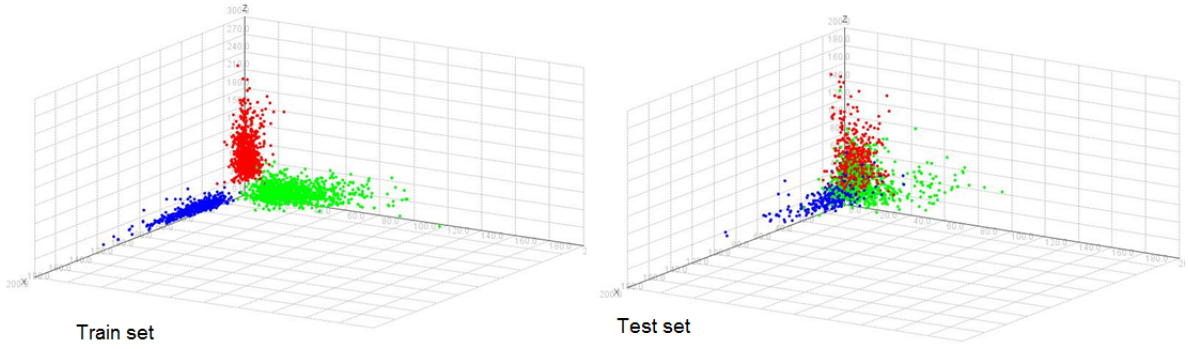

Figure 4.6: Document representation according to the novel TW: Games

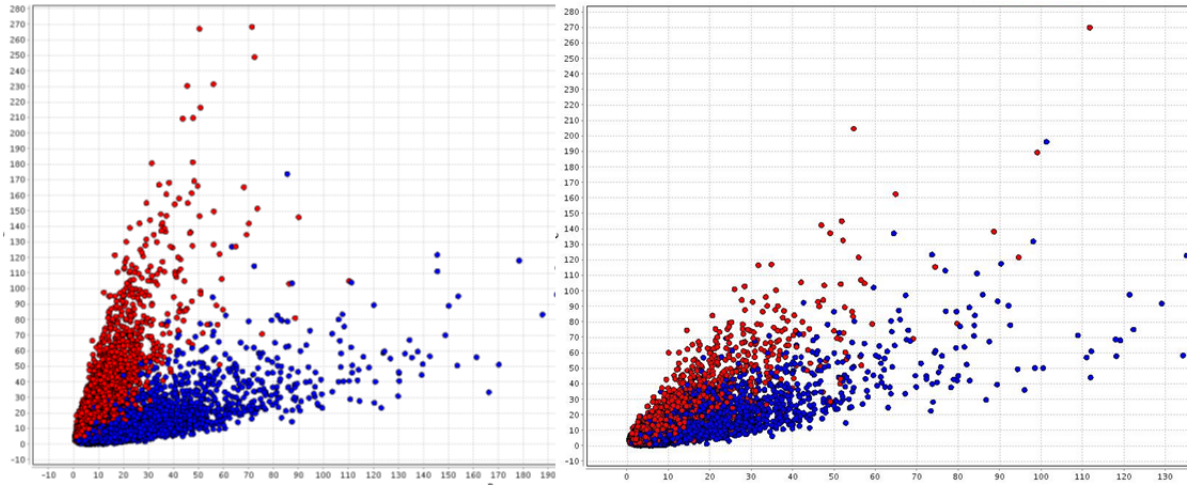

Figure 4.7: Document representation according to the novel TW: Debates

From Figure 4.7 one can see that in the case of the Debates corpus the objects of the training and the test set are not so different as in the case of Books and Games corpora.

Figure 4.8 illustrates the document representation obtained on the T1 corpus using the decision rule 4.3. This database contains four categories, Figure 4.8 shows projection of the four-dimensional feature space into three categories: Art, Economy and Sport.

The document representation obtained on the T2 corpus using the decision rule 4.3 is shown in Figure 4.9. Since the T2 corpus consists of five categories, Figure 4.9 provides a projection of the five-dimensional feature space into the three-dimensional subspace (France, International and Literature).

A problem with DEFT corpora (Books, Games, Debates, T1 and T2) is the size of the term set. It is difficult for most of the classification algorithms to deal with high dimensionality. In the following section a semi-supervised text preprocessing method is introduced.

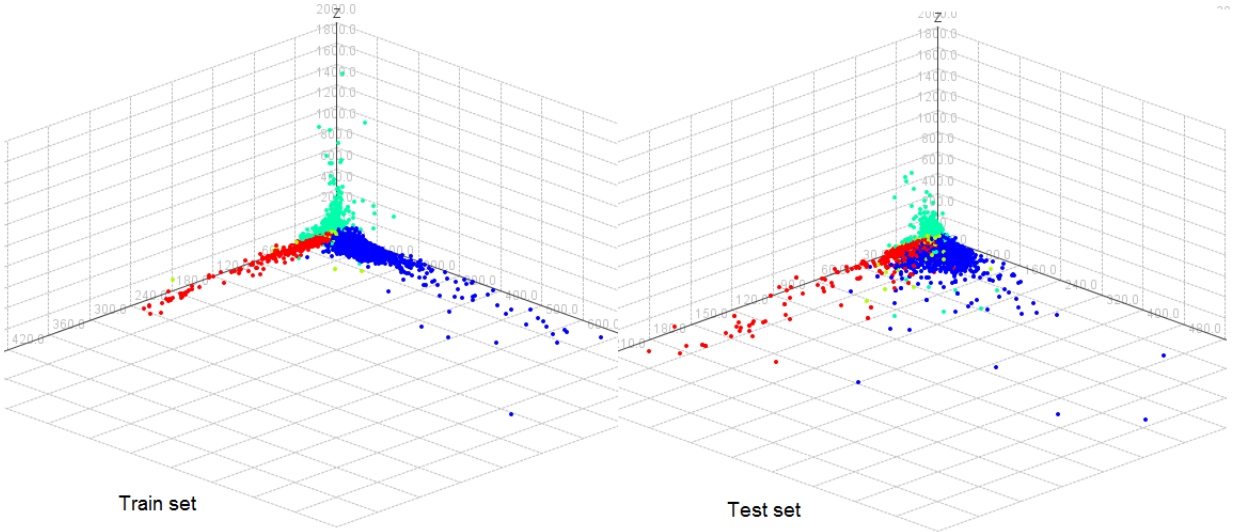

Figure 4.8: Document representation according to the novel TW: T1

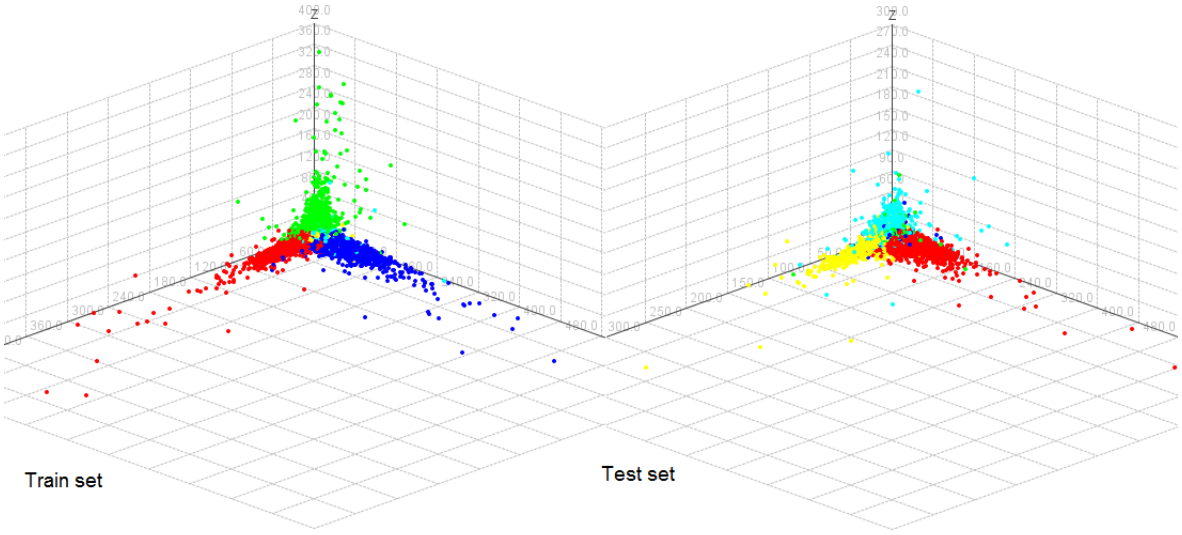

Figure 4.9: Document representation according to the novel TW: T2

4.1.3 Feature Space Dimensionality Reduction

This section introduces the developed methods of dimensionality reduction of the feature space. The details of this approach are described in Gasanova *et al.* (2014a). The dimensionality reduction step in text preprocessing is presented in Figure 4.10.

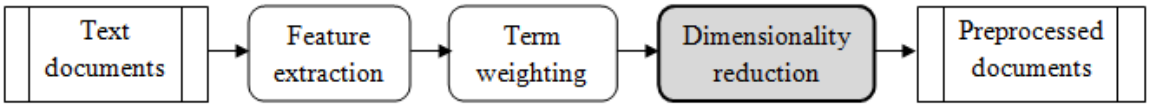

Figure 4.10: The dimensionality reduction step in text preprocessing

These algorithms have been designed for the corpora with large term set. In this work algorithms which reduce the initial term set have been applied only for DEFT 2007 and 2008 data sets and have not been used for SpeechCycle corpus. Since the SpeechCycle database has only about 3500 terms and many of these terms have an impact on the performance because people have known that they speak with automated agent and, therefore, calls are often just a set of key words.

4.1.3.1 Feature Selection

In this section a term filtering approach for dimensionality reduction of the feature space is introduced. Since many terms have close to zero weights they do not contribute significantly into classification process and increase CPU's computational time one can select a subset of features that represents the text documents with high enough resolution for classification algorithms to detect categories of text documents. Figure 4.11 illustrates the term filtering step in text preprocessing.

A term filtering algorithm is described as follows. Let $F_I = \{f_1, f_2, \ldots, f_N\}$ be the initial feature space that contains N features. Each feature f_i is associated with a weight w_i, where $1 \leq i \leq N$. The reduced feature space F_R is created according to the following rule.

For each f_i from the initial feature space F_I, where $1 \leq i \leq N$, the following holds

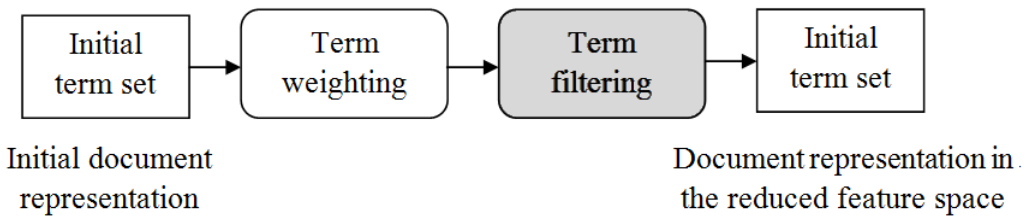

Figure 4.11: Term filtering step in text preprocessing

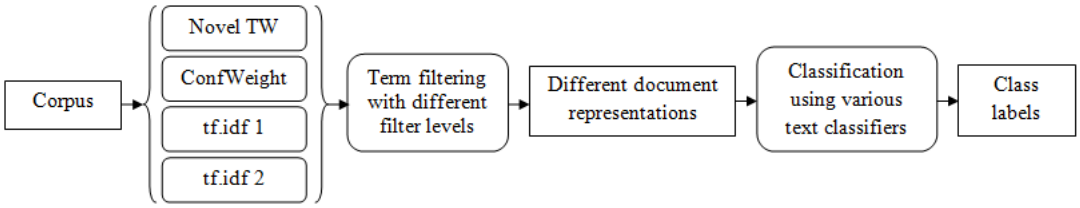

Figure 4.12: Term filtering approach

$$\begin{array}{rl} \text{if } w_i > \text{filter\_level, than} & f_i \in F_R, \\ \text{otherwise} & f_i \notin F_R. \end{array} \tag{4.4}$$

A filter level or a borderline value is defined as a real-valued parameter that is generally selected by experts. This value is highly depends on the applied term weighting method, the training data and the text classification task.

Since there is no deterministic method to find optimal borderline values as it highly depends on the training data. Term filtering with the chosen value can produce good results using one text preprocessing and does not perform well using other text preprocessing techniques. It is a complicated task to find a borderline to filter term weights. In this thesis the choice of an appropriate filter level is performed empirically by applying text classification algorithms using different term weighting formulas and randomly chosen borderline values. The performance depending on the filter level is estimated using the F-score performance measure. For each database a set of 10 filter levels is selected and compared with the performance obtained with the initial feature space.

For each corpus and every text preprocessing technique a set of the borderline values is selected and, then, a vector-space representation of the text collection is created. After that, using the training data preprocessed according to the text preprocessing algorithm and the applied filter level several classifiers are trained and, then, tested on the unlabelled data. The scheme of this approach is shown in Figure 4.12.

The described feature selection algorithm significantly reduces the dimensionality of the feature space, enabling to process and classify data using less CPU's computational time and the numerical experiments conducted in this thesis have shown better or similar performance using the reduced term set than the results obtained with the initial term set. The experimental setup, applied filter levels and results of the numerical experiments conducted in this thesis are presented in Chapter 5.

The term filtering approach is algorithmically straight forward, however, the problem of choosing the appropriate filter levels requires expert intuition and/or extensive numerical study, which makes it rather complicated, since it highly depends on human supervision. In

Algorithm 4.1 Proposed approach for semi-supervised dimensionality reduction

1. Choose the category for each term (where it appears more often than in others)

$$class\ winner_j = \max_{1 \le i \le n} \frac{|\{\text{documents from the class } i \text{which contain the term } j\}|}{|\{\text{all documents which belong to the class } i\}|}, \quad (4.5)$$

   where n is the number of classes.

2. Collect all terms which belong to the same class.
3. Create a dendrogram for each class by applying the hierarchical agglomerative clustering.
4. Choose the number of term clusters for each category.
5. For every term cluster calculate the common weight as an average weight of all elements in the cluster.
6. Optimize the common weights of the term clusters by cooperative scheme of the coevolutionary genetic algorithm: the set of term clusters for each category is considered as a subpopulation in GA.

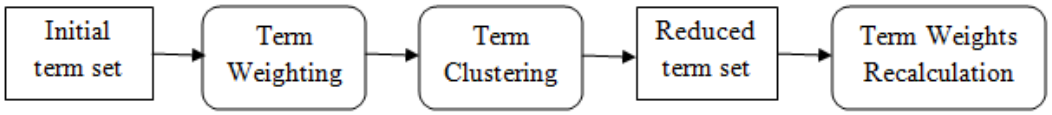

Figure 4.13: The scheme of the proposed dimensionality reduction approach

the next section a novel dimensionality reduction approach based on semi-supervised term clustering is introduced. It requires more computational time, however it does not depend on human supervision.

4.1.3.2 Feature Extraction

In this thesis a novel method for dimensionality reduction is proposed. It combines supervised and unsupervised learning model. The scheme of the proposed semi-supervised dimensionality reduction approach is described in Algorithm 4.1.

The introduced method for feature space dimensionality reduction is shown in Figure 4.13.

Genetic algorithms are well-known and widely used methods of global optimization since they can work with algorithmically defined criteria on the binary, real, nominal, mixed etc. data types, they search in parallel in different regions of the feature space. To provide local convergence genetic algorithms are often hybridized with methods of local optimization. Due to these features in our approach genetic algorithms combined with local search have been chosen as an optimization tool. In order to improve the accuracy rate using all words a

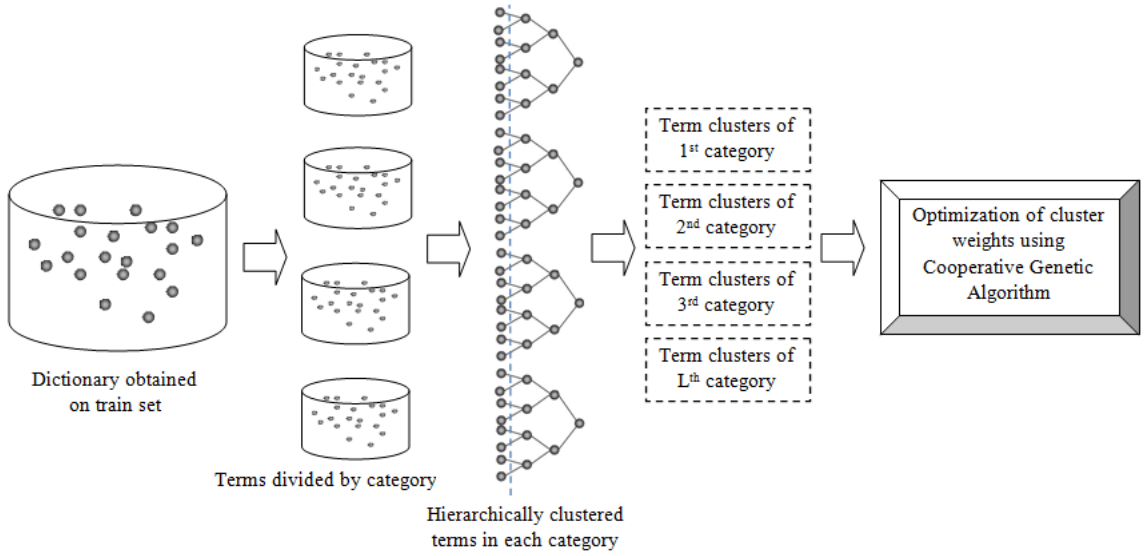

Figure 4.14: Block diagramm of the semi-supervised algorithm

coevolutionary genetic algorithm is proposed where individual algorithms do not compete with each other, but exchange the information between subpopulations.

Figure 4.14 shows the main stages of the proposed approach. First, for each word from the obtained term set the relative frequencies are calculated and the class with the highest relative frequency is assigned for this word. Then the feature space is divided into the groups of terms according to the assigned category. On the next step, the hierarchical agglomerative clustering algorithms are applied for each set of terms separately. Given the dendrograms for terms of all categories, the number of term clusters is chosen (it can be different for each of the categories). On this step, the cooperative scheme of the coevolutionary algorithm is applied to optimize weights of the chosen term clusters.

The first step of the proposed approach is to divide extracted term according to the category, where it reaches the maximum of the relative frequencies, and to perform term clustering for each group of terms separately. In the following section this step is described in detail.

Term Clustering

In this section a term clustering approach for dimensionality reduction of the feature space is introduced. The motivation of using term clustering instead of any of the term selection strategies such as term filtering is clear. Despite of the computational simplicity such filtering is sensitive to the value which is used as a borderline for term weights. The aim of this work is an attempt to eliminate as much as possible the human supervision and parameters settings

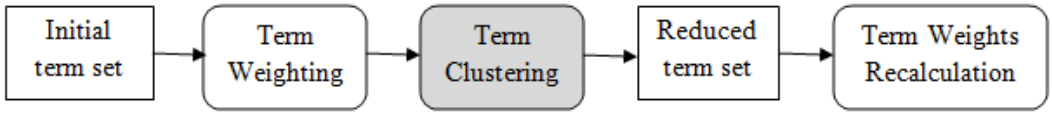

Figure 4.15: Term clustering step in text preprocessing

Category	Initial term set	**Reduced term set**
Positive	11714	1123
Neutral	15071	661
Negative	25722	1078
Total	*52507*	***2862***

Table 4.2: Initial term set extracted from the training data and reduced term set after grouping terms with identical weights: Books

from the text preprocessing. Figure 4.15 shows the term clustering step in text preprocessing.

For each word extracted from the train data the corresponding weight and the number of class where it contributes have been assigned. Due to the size of the term set it is time consuming and nontrivial to directly apply any optimization method. Therefore, the initial term set should be processed in the way that terms that have equal or similar weights be in the same cluster and one common weight will be assigned to all terms from the cluster.

It should be mentioned that our preprocessing stage does not use any specific linguistic information, expert knowledge or domain related information. Therefore, it can be easily transportable to the data from another domain or even in another language.

Terms that have the same weight and belong to one category can be considered as the same term. Tables 4.2, 4.3, 4.4, 4.5 and 4.6 show how dramatically the feature space dimensionality decreases if terms with identical weights are grouped into term clusters. Corpora of the DEFT 2008: T1 and T2 have the highest feature space dimensionality even after the uniting terms with the same weights.

However, from Tables 4.2, 4.3, 4.4, 4.5 and 4.6 one can see that the number of clusters with unique term weights is still too large for an optimization method. It is a complicated task to optimize over 30000 variables and, therefore, in order to reduce the feature space dimension

Category	Initial term set	**Reduced term set**
Positive	16418	2083
Neutral	21309	1489
Negative	25417	2940
Total	*63144*	***6512***

Table 4.3: Initial term set extracted from the training data and reduced term set after grouping terms with identical weights: Games

Category	Initial term set	**Reduced term set**
For	30752	1497
Against	28863	1926
Total	*59615*	***3423***

Table 4.4: Initial term set extracted from the training data and reduced term set after grouping terms with identical weights: Debates

Category	Initial term set	**Reduced term set**
Sports	45471	6503
Economy	52838	8476
Art	80052	8027
Television	24618	6945
Total	*202979*	***29951***

Table 4.5: Initial term set extracted from the training data and reduced term set after grouping terms with identical weights: T1

Category	Initial term set	**Reduced term set**
France	29145	5192
International	46102	7238
Literature	65919	8068
Science	80412	5760
Society	40822	4793
Total	*262400*	***31051***

Table 4.6: Initial term set extracted from the training data and reduced term set after grouping terms with identical weights: T2

hierarchical agglomerative clustering algorithm is taken.

Hierarchical Agglomerative Clustering

The choice of the clustering algorithm depends on the purpose. In this case the hierarchical agglomerative clustering algorithm has been selected since it has an important advantage: the output is the hierarchical structure of terms and there is a possibility to choose different numbers of term clusters without the need to repeat the calculations.

In order to reduce the feature space dimensionality the clustering algorithm has been applied to terms of each category separately.

All hierarchical algorithms differ in the way of measuring the distance between clusters. In this case the Euclidean distance has been chosen as the most natural one since the elements of clusters are real-valued term weights.

To choose which clusters are joint on the current step all distances between clusters are calculated. Average-linkage clustering algorithm has been used.

The distance between cluster X and Y is given by

$$dist(X,Y) = \frac{1}{N}\frac{1}{M}\sum_i\sum_j ||X_i - Y_j||, \tag{4.6}$$

where N is the number of words in cluster X and M is the number of words in cluster Y; and the closest clusters are united.

Algorithm 4.2 shows the common scheme of the average-linkage clustering.

Algorithm 4.2 Average-linkage clustering

1. First all terms are considered as clusters contained just one element.
2. All distances between clusters are calculated and all pairs of clusters which have the smallest distance calculated using Equation 4.6 are united.
3. Then the distances between new clusters and other ones are evaluated and process of combining term clusters go on untill all terms belong to one cluster.

An output of the clustering algorithm is the dendrogram of terms.

As a common term cluster weight W_i of the cluster i the arithmetic mean of all term weights from the cluster i is calculated. W_i is given by

$$W_i = \frac{1}{N_i}\sum_{j=1}^{N_i} w_{ij}, \tag{4.7}$$

where N_i is the number of terms in the ith cluster and w_{ij} is a weight of the jth term from the ith cluster.

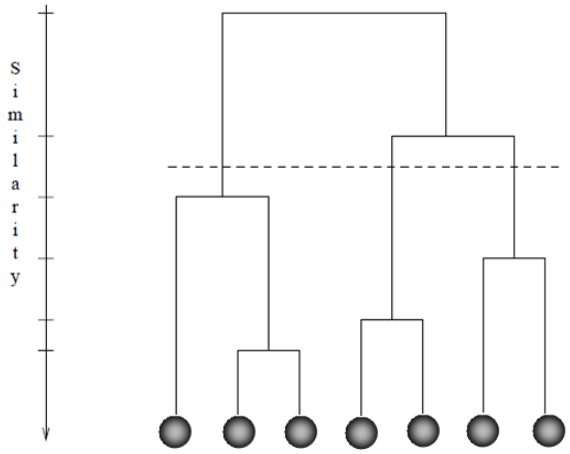

Figure 4.16: Hierarchical Agglomerative Clustering

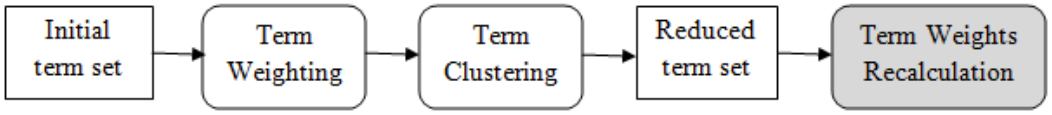

Figure 4.17: Term weights recalculation step in text preprocessing

There is no evidence that these common weights will provide the best possible classification. Therefore, in the following section the optimization of these common weights of the obtained term clusters is proposed.

4.1.3.3 Optimization of the Term Cluster Weights

This section describes the optimization task of common term clusters weights and cooperative scheme of coevolutionary genetic algorithm applied for this problem. First, the mathematical statement of the optimization problem is presented, then the standard evolutionary algorithm and genetic operators such as selection, crossover and mutation are described. Finally, the cooperative scheme used in this work is explained in detail. A semi-supervised modification of the fitness function in order to overcome the overfitting problem are also discussed.

Figure 4.17 shows term weights recalculation step in text preprocessing.

Optimization Problem Statement

Given the hierarchical trees for each category contained terms as leaves (terminal nodes) and chosen number of term clusters (i.e. where the tree is cut), our task is to find optimal weights of the term clusters for all categories.

Let K be the number of categories, C_i is the ith category, S_i is the number of term clusters

that belong to ith category and $n = \{n_1, \ldots, n_K\}$ is the vector of chosen term cluster numbers for each of the categories $C_i = \{t_{i1}, t_{i2}, \ldots, t_{iS_i}\}$.

The set of weights of term clusters is given by the matrix

$$W = \left\{ \begin{array}{cccc} (w_{11} & w_{12} & \ldots & w_{1n_1}) \\ (w_{21} & w_{22} & \ldots & w_{2n_2}) \\ \ldots & \ldots & \ldots & \ldots \\ (w_{K1} & w_{K2} & \ldots & w_{Kn_K}) \end{array} \right\}, \tag{4.8}$$

where w_{ij} is a weight of term cluster j in ith category.

The document d_l is given by the vector

$$d_l = \{t_{l1}, t_{l2}, \ldots, t_{lM_l}\}_l, 1 \leq l \leq N, \tag{4.9}$$

where N is the number of documents in the training set and M_l is the number of terms in the d_l document.

For each document d_l the vector of sums of weights is calculated using the following formula:

$$\left(\sum_{i=1, t_{li} \in C_1}^{M_l} w_{1i}, \ldots, \sum_{i=1, t_{li} \in C_K}^{M_l} w_{Ki} \right). \tag{4.10}$$

The class assigned to the document d_l is defined as

$$class_l = \arg\max_k \sum_{i=1, t_{li} \in C_k}^{M_l} w_{ki}. \tag{4.11}$$

The optimization task is determined as a maximization of an accuracy rate obtained on all documents from the training set. It is given by

$$\frac{1}{N} \sum_{l=1}^{N} \delta_l \to \max, \tag{4.12}$$

where δ_l is defined as

$$\delta_l = \begin{cases} 1 \text{ if } class_l = label_l, \\ 0 \text{ otherwise} \end{cases}. \tag{4.13}$$

Since there is not enough information about the behaviour of the objective function, heuristic optimization methods such as evolutionary algorithms can be useful.

Genetic Algorithm Applied for Optimum Term Clusters Weighting

Each individual is considered as a set of weights of term clusters for one of the categories which are encoded in the binary sequence. In this work in order to classify the object to one of the given categories the decision rule 4.3 is applied due to its computational simplicity. Since text collections contain thousands of documents and in order to estimate a fitness value of a candidate solution, all text documents from the training data have to be classified, CPU's computational time required to assign a class label to one document has to be minimized.

The reason that genetic algorithms cannot be considered to be a lazy way of performing design work is precisely because of the effort involved in designing a workable fitness function. Even though it is no longer the human designer, but the computer, that comes up with the final design, it is the human designer who has to design the fitness function. If this is designed badly, the algorithm will either converge on an inappropriate solution, or will have difficulty converging at all.

Moreover, the fitness function must not only correlate closely with the designer's goal, it must also be computed quickly. Speed of execution is very important, as a typical genetic algorithm must be iterated many times in order to produce a usable result for a non-trivial problem.

An accuracy rate obtained on the training set is used as a fitness function (optimization criterion).

$$\text{fitness function} = \frac{\text{Number of correctly classified objects}}{\text{Number of all objects}} \tag{4.14}$$

To provide local convergence genetic algorithms are often hybridized with methods of local optimization. Due to these features in our approach genetic algorithms combined with local search have been chosen as an optimization tool. In this work the local area of the best obtained individual on each generation is searched for better solutions. The reason to perform the local search only on the best individual in every generation is to improve the algorithm performance without too much loss in computational efficiency.

Algorithm 4.3 describes the scheme of local search.

Algorithm 4.3 Local search

1. Set the counter $m = 0$.
2. Save the best value of the objective function F_{best}.
3. Randomly choose the position in chromosome l.
4. Invert the l bit.
5. Calculate the value of the objective function F_{new}.
6. If $F_{new} > F_{best}$ than save the change of l bit and set $F_{best} = F_{new}$.
7. Increment the counter $m = m + 1$.
8. If $m > M$ than END; otherwise go to step 3.

In order to improve the accuracy rate a coevolutionary genetic algorithm is proposed, where individual algorithms do not compete with each other, but exchange the information between subpopulations. After terms are clustered in the dictionary there is a hierarchical tree for each category and assigned values to all clusters. The question if these values are a global optimum remains open. There is no evidence that the current values are even a local maximum of classification quality function.

In order to optimize weights of the obtained term clusters a genetic algorithm hybridized with local search is applied, due to its relative simplicity and global convergence, and it does not require any a priori information about behaviour of the classification quality function.

In this thesis a local search algorithm is applied only to the best obtained individual to make sure that it reaches at least a local maximum. The weights of term clusters of other categories are fixed and only the weights of term clusters of the current class are being optimized. Each individual represents weights of term clusters for the current category encoded to a binary string. The accuracy on the training set is used as a fitness function. It is calculated with the fixed weights of term clusters and weights of term clusters of the individual.

Cooperative Scheme of Coevolutionary Genetic Algorithm

In contrast with the evolution concept which is naturally applied to the single-objective optimization problems of different complexity, the coevolution decomposes the task implicitly or explicitly in order to take advantages with complex problem which can be divided into the simpler ones. It is logically to assume that this coevolution can achieve better results dealing with the parts of problems independently rather that the evolutionary algorithms which evolve the entire structure.

Topic categorization tasks as described above can be easily decomposed into K subtasks

(K is the number of categories). Each of this subtasks is represented the optimization of weights of term clusters which belong to one category assuming that other term cluster weights are fixed.

Therefore, each subtask i can be formulated as: for every document d_l from the training set find such vector of weights $W_i = (w_{i1}, w_{i2}, \ldots, w_{in_i})$ that maximizes the accuracy obtained with this vector and fixed weight vector $W_j, j \neq i$.

The best obtained weights of term clusters which belong to other categories are fixed.

Cooperative scheme of the coevolutionary genetic algorithm assumes that the individuals of all subpopulations do not compete with each other and do not migrate from one subpopulation to another (in this settings individuals of different subpopulation cannot be directly compared because chromosomes are represented different values), but share the obtained information which is used to calculate the fitness functions. In the proposed application subpopulations have periods when they work independently knowing only the best values shared in the beginning. Then the evolution stops and the best term cluster weights are updated.

Each of the individual genetic algorithms are hybridized with the local search that is applied only to the best individual obtained in every generation.

In order to take advantage of the improvement of all term cluster weights an application of the cooperative coevolutionary genetic algorithm with local search is proposed. Local search is applied randomly m times only to the best individual on each generation.

The main stages of applied method are shown in Figure 4.18.

On the first phase all individual genetic algorithms work separately (for each of them other term cluster weights are fixed and the task is to optimize term cluster weights which belong to the corresponding class), the length of this phase defines how many generations individual algorithms can work without information exchange. Then all separate algorithms are stopped and update all term cluster weights which have been obtained by the best individuals of all algorithms. These two stages are repeated until the maximum number of generations is exceeded.

This variant of coevolutionary algorithm uses a cooperative scheme in order to achieve higher performance than each individual algorithm, in this case subpopulations do not exchange individuals, only information that influences the fitness function calculation.

Algorithm 4.4 describes the scheme of the cooperative scheme of the coevolutionary genetic algorithm.

The overall diagram of the proposed text preprocessing method is shown in Figure 4.19.

Semi-Supervised Modification of Fitness Function

The optimization of the classification model becomes a difficult problem in Books and Games corpora due to the significant difference between the training and the test data and an over-

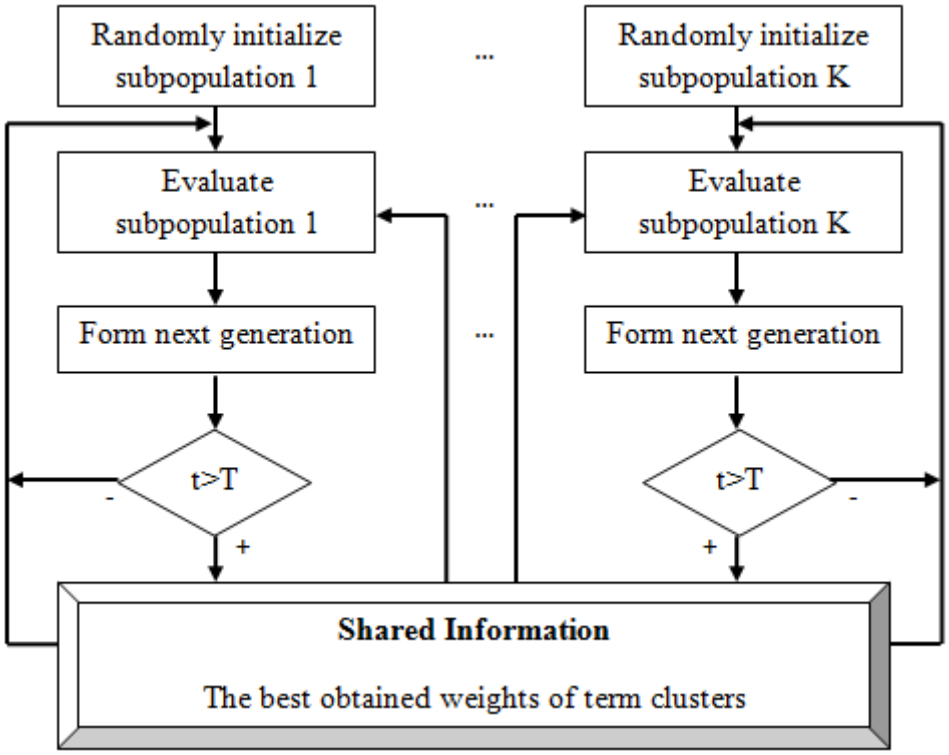

Figure 4.18: Cooperative scheme of the coevolutionary genetic algorithm for term cluster weights optimization

fitting problem. In a short time accuracy rate (or F-score) on the training set reaches over 0.95 and cannot improve the classification quality on the test set any further.

The objective function can be modified in the way that it will provide better partitions into classes with the same classification quality on the training set, which gives the possibility to obtain better results on the test set. Therefore, the optimization criterion (the fitness function of the genetic algorithm) has to be changed.

The sum of existing supervised and additional unsupervised criteria is proposed to be the new fitness function. This modification is applied only to the individuals that have reached over 0.95 accuracy on the training set, since it is pointless before the very last stage when the supervised criterion is more important.

The modified fitness function is given by

$$\text{fitness} = \begin{cases} \text{accuracy}_{train} & \text{if accuracy}_{train} < 0.95 \\ \text{accuracy}_{train} + \text{clustering criterion} & \text{otherwise} \end{cases} \rightarrow \max . \qquad (4.15)$$

Algorithm 4.4 Cooperative scheme of coevolutionary genetic algorithm

1. Save the common weights of all term clusters (they will be used to calculate fitness function) and set the iteration number $g = 0$.
2. Randomly initialize all subpopulations.
3. Set the current number of subpopulation $i = 1$ and the time $t = 0$.
4. Calculate the fitness functions of the i subpopulation: for each individual the encoded weights of term clusters in i subpopulation are used with the obtained best common weights of the term clusters in all other subpopulations (or if $g = 0$ than the saved common weights calculated in the step 1).
5. Perform the local search on the best obtained individual.
6. Save the best value of fitness function.
7. If $t > T$ (T is the number of generations when subpopulations can work separately) than go to step 9.
8. Apply genetic operators: selection, crossover, mutation and form a next subpopulation. Set $t = t + 1$. go to step 4.
9. Set $i = i + 1$ and the time $t = 0$.
10. If $i > K$ (K is number of classes in the classification task) than $g = g + 1$ and go to step 11. Else go to step 4.
11. If $g > G$ (G is the maximal number of generations, stopping criterion) than END. Else go to step 3.

It should be noted that in this case it is meaningless to use the external evaluation criteria since the fitness function has been already optimized the accuracy on the training set which is the external criterion with the use of known class labels.

There are a lot of internal criteria to evaluate the quality of the obtained cluster partition, however, for the purpose of the genetic algorithm the criterion has to be computed fast enough to be an effective fitness function. Due to its relative computational simplicity maximization the average distance between centres of obtained representation vectors (low inter-cluster similarity) has been chosen as an internal criterion.

This internal criterion is given by

$$\text{clustering\_criterion} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \|c_i - c_j\| \rightarrow \max, \tag{4.16}$$

where N is the number of clusters, c_i is a centroid of the cluster i and c_j is a centroid of the

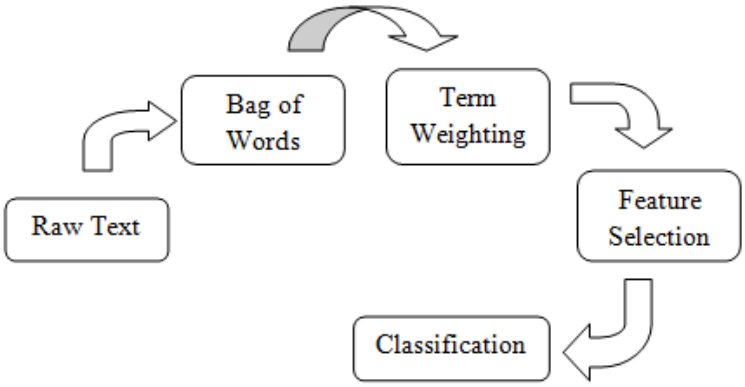

Figure 4.19: The proposed text preprocessing

cluster j.

Using the combination of the supervised and unsupervised learning provides the class partitions which are the most separated from each other. However, the obtained clustering should still be good enough according to the supervised criterion.

It should be noted that in this criterion the objects with true labels and predicted labels are both used to calculate the unsupervised objective which is the additional information which helps to provide better classification results.

This modification allows us to improve the performance of the classification algorithms using the obtained representation up to 1-2% according to Amax decision rule.

4.1.4 Clustering Non-Uniform Classes

This section describes a semi-supervised classification algorithm that uses unsupervised learning to improve results of supervised classification algorithm. The details of the proposed approach are described in Gasanova *et al.* (2013a,c).

This approach is based on the assumption that it is easier to classify objects that belong to the relatively small well-defined classes with uniform structure than objects from one class that contains different types of elements. The method introduced in this section has been designed to improve performance on the SpeechCycle corpus, since this dataset has one special category that is difficult to detect.

After the analysis of the performances of standard classification algorithms on the given database, it has been concluded that there exists one specific class (class _TE_NOMATCH), where all standard techniques perform worse. Due to the non-uniform structure of this class it is difficult to detect the whole class by classification methods. If decision rule 4.3 is applied directly, only 55% of accuracy rate is achieved on the test data (61% on the train data). This thesis suggests to divide the _TE_NOMATCH class into a set of sub-classes using one of the

Parameter	Value
Population size	50
Number of generations	50
Number of runs	50
Number of bits to code each variable	7
Variables lie in the interval	[0;1]
Step	0,0078125
Selection	Tournament with the size = 3
Crossover	Uniform
Mutation	$Probability = \frac{1}{Chromosome\,length}$

Table 4.7: Settings of the genetic algorithm applied for clustering of the non-uniform class

clustering methods and, then, recount the term weights C_j taking into account that there are other informative categories and that the set of the well-defined subclasses are considered as separate classes.

In this work the following clustering methods are used: genetic algorithm with integers, vector quantization network trained by a genetic algorithm, hierarchical agglomerative clustering with different metrics and linkage criteria.

4.1.4.1 Genetic Algorithm with Integers

Since genetic algorithms are well-known heuristic for the optimization of the complex functions without a-priory knowledge about their behaviour on the search space, in this work the simplest idea is to directly apply GA to the clustering task.

Each individual is represented as a sequence of nonnegative integer numbers (each number corresponds to the number of _TE_NOMATCH subclass). The length of this sequence is the number of utterances from the training set which belong to the _TE_NOMATCH class. The training set accuracy is used as a fitness function, which means that the quality of the obtained clustering of the part of the database is measured as the classification result obtained on the whole database. This genetic algorithm is applied to find directly the optimal clustering using different numbers of clusters and it can be concluded that with increasing the clusters number (in the _TE_NOMATCH class) the better classification accuracy on the whole database is obtained.

The parameters of GA that have been used in the numerical experiments are presented in Table 4.7.

Applying this method about 7% improvement of accuracy rate on train data and about 5% on test data is achieved. The numerical results are shown in Figure 4.20. One can notice that

after about 35 subclasses further increase in number of clusters does not produce significant improvement in the accuracy rate.

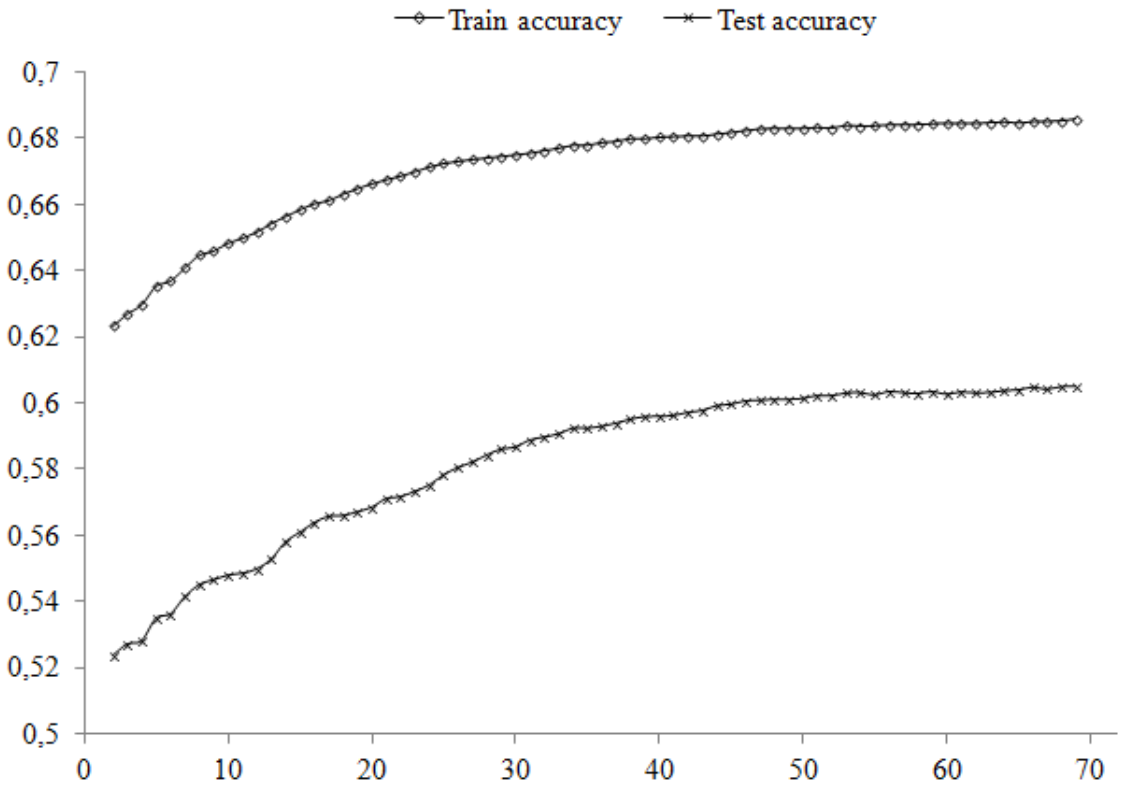

Figure 4.20: Numerical experiments conducted using GA (Ox indicates a number of _TE_NOMATCH subclasses; Oy indicates an accuracy obtained on the training and test data)

4.1.4.2 Learning Vector Quantization trained by Genetic Algorithm

In this thesis a learning vector quantization (LVQ) algorithm has been also implemented using Microsoft Visual Studio C++.

For the given number of subclasses of the non-uniform class a set of code vectors (the number of code vectors is equal to the number of subclasses) is defined. These code vectors are optimized using a genetic algorithm, where as a fitness function the classification accuracy on the training set is used. Each code vector corresponds to the certain _TE_NOMATCH subclass. The object belongs to the subclass if the distance between it and the corresponding code vector is smaller than the distances between the object and all other code vectors. In this thesis cosine similarity is used as a distance function. Figure 4.21 illustrates LVQ algorithm.

The following parameters of GA have been used: population size = 50, number of genera-

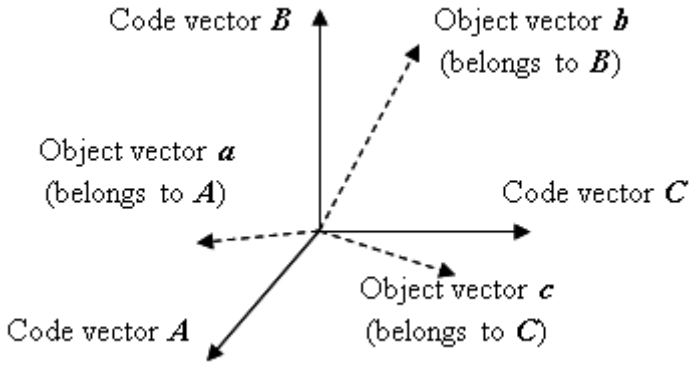

Figure 4.21: LVQ code vectors

tion = 50, mutation probability = 1/chromosome_length, tournament selection (tournament size = 3), uniform crossover, averaged by 50 runs. Applying this algorithm to the given database the similar results as in the case of genetic algorithm with integers have been obtained. Figure 4.22 illustrates numerical results obtained with LVQ algorithm on SpeechCycle corpus.

4.1.4.3 Hierarchical Agglomerative Clustering

In this thesis hierarchical agglomerative binary clustering has also been applied, where each subclass contains only one utterance and, then, classes are consequently grouped in pairs until there is only one class containing all utterances or until the certain number of classes is achieved. The performance of hierarchical clustering algorithms depends on the metric (the way to calculate the distance between objects) and the criterion for clusters union. In this thesis the Hamming metric with single-linkage, complete-linkage, average-linkage and Ward criterion have been used. The best performance has been achieved with the Hamming metric and Ward criterion.

Hamming distance between two sequences of equal length is calculated as the number of positions where the symbols are different. It can be considered as a number of substitutions (or errors) required to transform first sequence to the second one. Ward criterion has been described in Chapter 2.3.2.1.

Number of subclasses varies from 2 to 100. The numerical experiments have been carried out in GNU project R which is generally used for statistics and data mining.

Linkage criteria used in simulations are:

- Single-linkage
  Results achieved using single linkage are presented in Figure 4.23.

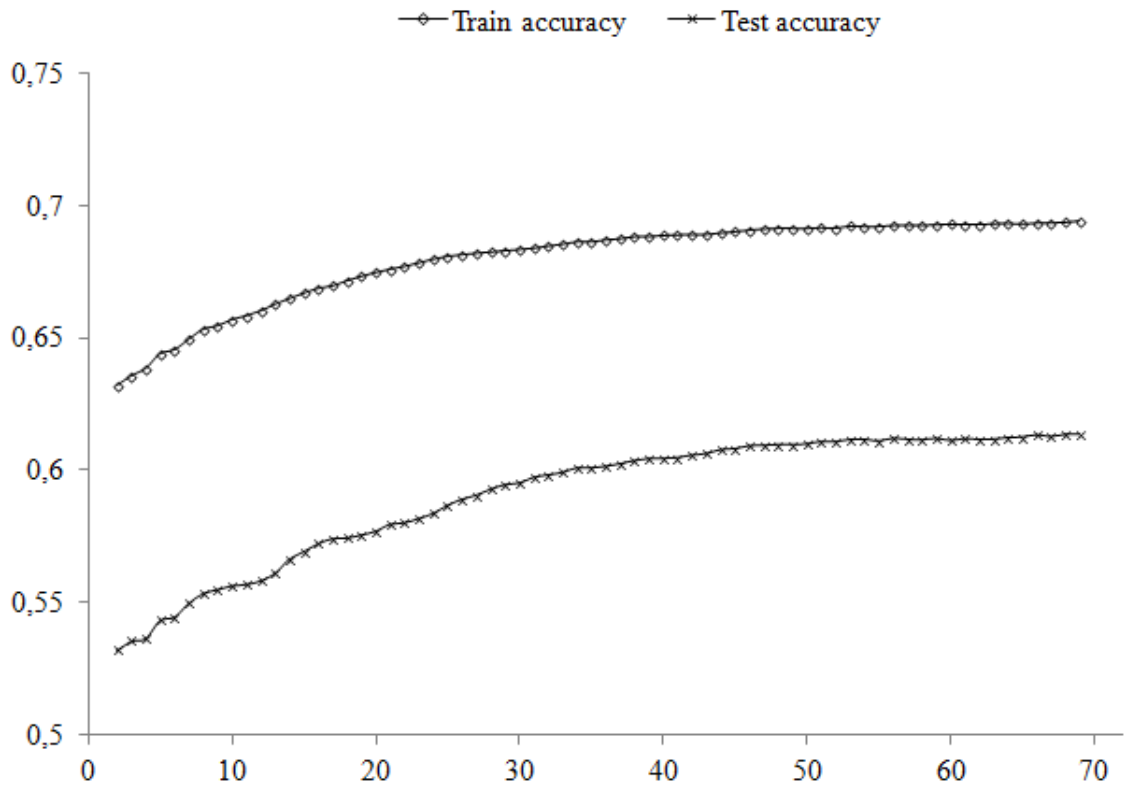

Figure 4.22: Numerical experiments conducted using GA (Ox indicates a number of _TE_NOMATCH subclasses; Oy indicates an accuracy obtained on the training and test data)

The best performance using single-linkage on the training data has been achieved using 33 subclasses with $Q_{train} = 64.12$ and the corresponding values obtained on the test set $Q_{test} = 58.70$.

- Complete-linkage
  Results achieved using complete linkage are presented in Figure 4.24.
  The best performance using complete-linkage on the training data has been achieved using 8 subclasses with $Q_{train} = 64.11$ and the corresponding values obtained on the test set $Q_{test} = 57.12$.

- Average-linkage
  Results achieved using average linkage are presented in Figure 4.25.
  The best performance using complete-linkage on the training data has been achieved using 15 subclasses with $Q_{train} = 64.99$ and the corresponding values obtained on the test set $Q_{test} = 59.47$, which outperforms the results with single- and complete-linkage

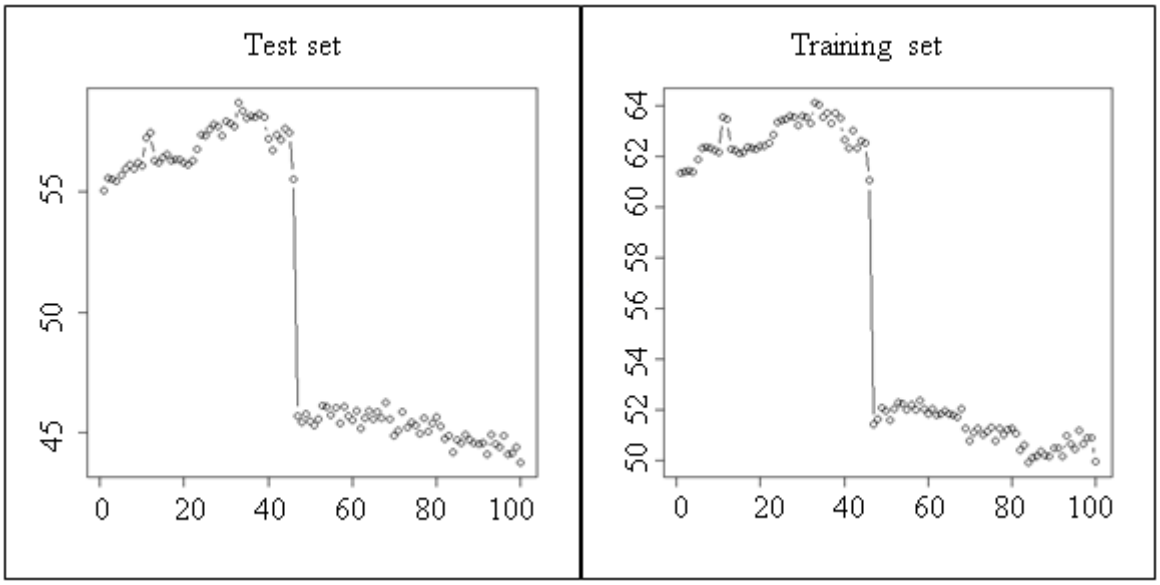

Figure 4.23: Numerical experiments conducted on the training and test sets using hierarchical agglomerative clustering (single-linkage) with different number of subclasses (x-axis represents the number of subclasses, y-axis represents accuracy rate)

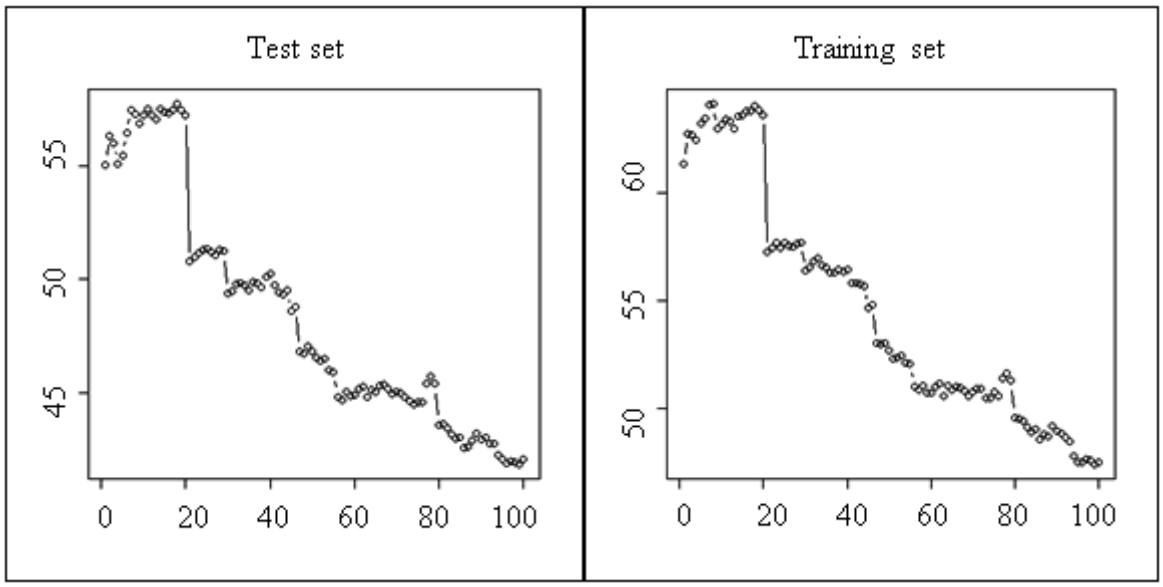

Figure 4.24: Numerical experiments conducted on the training and test sets using hierarchical agglomerative clustering (complete-linkage) with different number of subclasses (x-axis represents the number of subclasses, y-axis represents accuracy rate)

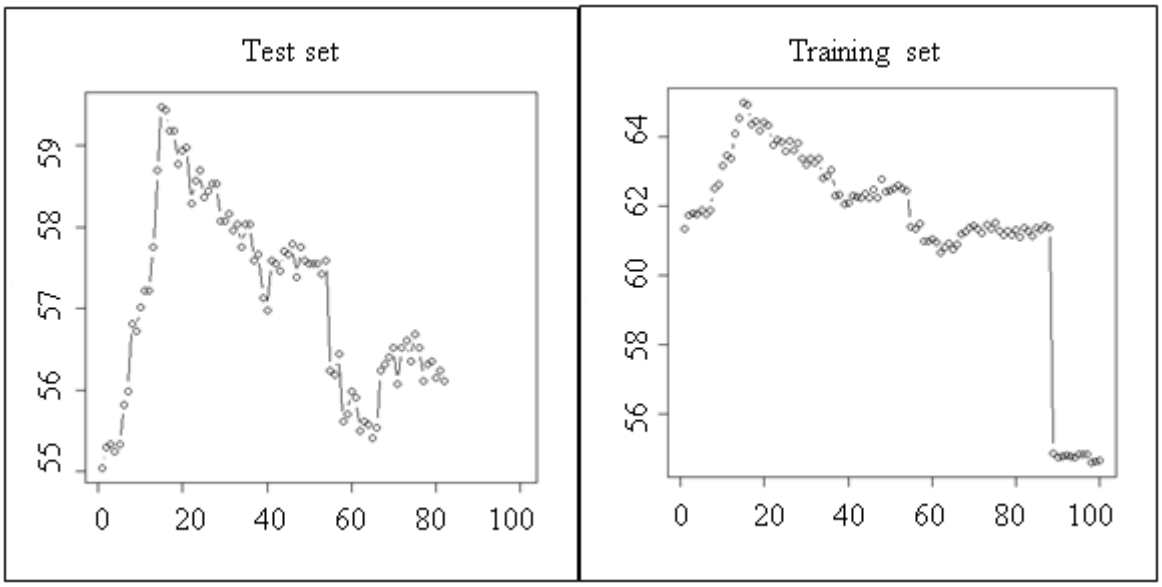

Figure 4.25: Numerical experiments conducted on the training and test sets using hierarchical agglomerative clustering (average-linkage) with different number of subclasses (x-axis represents the number of subclasses, y-axis represents accuracy rate)

clustering.

- Ward's criterion
  Results achieved using Ward's criterion are presented in Figure 4.26.
  The best performance using complete linkage on the training data has been achieved using 35 subclasses with $Q_{train} = 68.67$ and the corresponding values obtained on the test set $Q_{test} = 63.70$, which is the best obtained improvement. It should be mentioned that the best performance on the test set reaches $Q_{test} = 63.90$ using 45 subclasses.

The numerical experiments conducted in this thesis have shown that hierarchical clustering with Ward's criterion outperforms single-, complete- and average-linkage clustering algorithms on SpeechCycle corpus. Therefore, clustering the class of non-uniform objects can improve the classification quality using the same classification methods. Clustering with Ward's criterion improves the classification result on SpeechCycle corpus up to 8%.

Using the agglomerative hierarchical clustering about 9% improvement has been achieved. The best classification quality has been obtained with 35 subclasses on the training data (68.7%) and 45 subclasses on the test data (63.9%). Clustering into 35 subclasses gives 63.7% of accuracy rate on the test data.

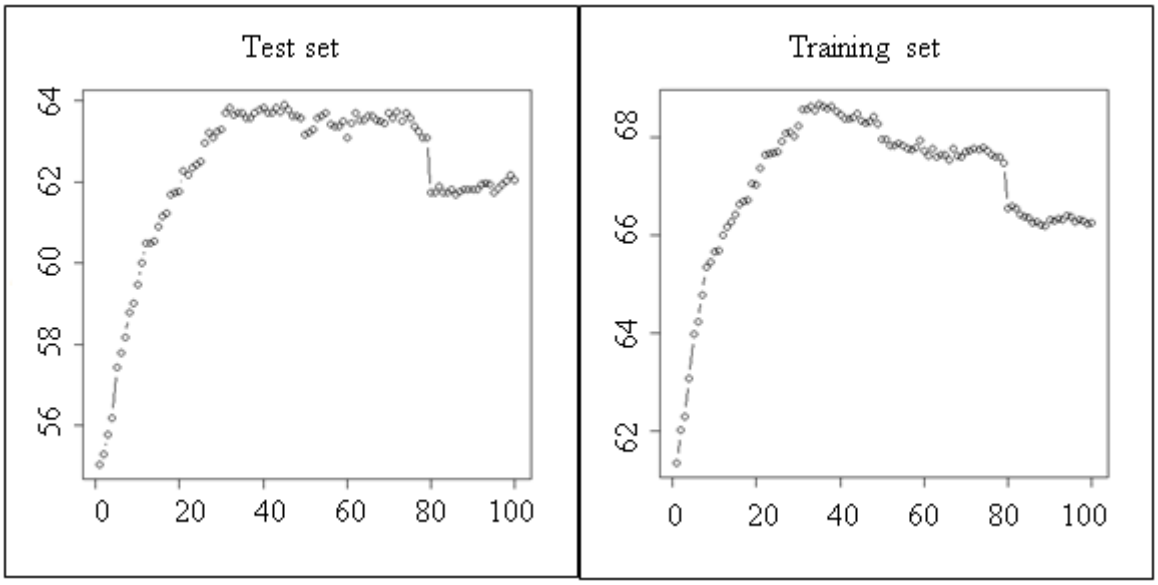

Figure 4.26: Numerical experiments conducted on the training and test sets using hierarchical agglomerative clustering (Ward's criterion) with different number of subclasses (x-axis represents the number of subclasses, y-axis represents accuracy rate)

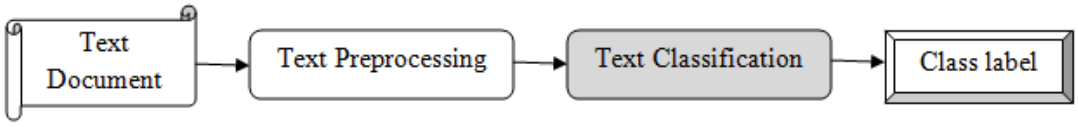

Figure 4.27: Classification step of text preprocessing and text classification

4.2 Novel Text Classification Methods

4.2.1 Introduction

The task of document categorization consists of text preprocessing and classification parts. In previous section text preprocessing techniques have been discussed in detail including a novel term weighting method and a novel dimensionality reduction algorithm. This section is concerned with classification of the preprocessed documents. Figure 4.27 presents the classification step of the text classification process.

In Joachims (1998) text categorization has been defined as a process of classifying text documents into a fixed number of predefined categories or classes. Text categorization has various applications, for example, automated indexing of scientific articles, identification of document genre, spam filtering and etc.

There are different methods for text categorization (statistical methods, rule-based methods and their combinations). This thesis focuses on the statistical approach, since it requires

significantly less human supervision and expert knowledge.

This chapter describes the generation and parameters optimization of the well-known text classifiers: Support Vector Machine (SVM) and Artificial Neural Network (ANN) using a metaheuristic algorithm called Co-Operation of Biology Related Algorithms (COBRA).

COBRA is a metaheuristic optimization method, which has been proposed by Akhmedova & Semenkin (2013). It is based on cooperation of biology inspired algorithms such as Particle Swarm Optimization (Kennedy *et al.* (1995)), Wolf Pack Search Algorithm (Yang *et al.* (2007)), Firefly Algorithm (Yang (2009)), Cuckoo Search Algorithm (Yang & Deb (2009)) and Bat Algorithm (Yang (2010)).

Support Vector Machine (SVM) has been proposed in Vapnik & Chervonenkis (1974) and has been shown good performance on the variety of machine learning applications including the text classification. In machine learning, the SVM is a supervised learning model; the basic SVM takes a set of input data and predicts which of possible classes forms the output. The SVM model is a representation of the examples (input data) as points in a space, mapped so that the examples of the separate classes are divided by a clear gap that should be as wide as possible. New examples are then mapped into the same space and are predicted to be belonging to a category dependently on which side of the gap they fall on.

This approach has been originally developed as a linear classifier (in the simplest terms, a hyper-plane that represents the largest separation between different classes is developed). However, it is often not possible to separate classes linearly. Boser *et al.* (1992) have suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes. In this thesis the polynomial kernel function with three parameters has been chosen for solving text categorization problems.

The common problem of the SVM-based classifier is the optimal choice of the parameters, which makes it difficult for a non-expert to apply this algorithm with a satisfying performance. In this chapter, the metaheuristic algorithm is applied to automatically find the optimal parameters of the SVM. In this work for generating the SVM structure the original COBRA is used: each individual in all populations represents a set of kernel function's parameters. Then for each individual constrained modification of COBRA is applied for finding vector w and the shift factor b. Finally, the individual with the best classification rate is chosen as the designed classifier.

Another popular method for the text classification is an Artificial Neural Network (ANN). If the structure (the number of layers, the number of neurons on each layer and the type of activation function in each neuron) and weight coefficients have been carefully selected, this approach produces good classification results. However, as in the case of SVM, it is a nontrivial problem to find the combination of the ANN parameters that gives an optimum.

In this thesis the structure of ANN is considered to be a binary matrix with N rows

(N is maximal number of neurons in each hidden layer) and M columns (M is a number of hidden layers). An optimization method designed for problems with binary variables (binary COBRA) is used to find the best structure of ANN and a real-valued optimization method (original COBRA) is applied to adjust weight coefficients for every structure.

In this chapter the applications of the metaheuristic approaches for real-valued optimization, for binary-valued optimization and for constrained optimization that generate and tune the SVM-classifier and generate structure and optimize parameters of the ANN are introduced.

4.2.2 Optimization of SVM and ANN-based classifiers with Co-Operation of Biology Related Algorithms

In this section structure generation and parameters adjustment of Artificial Neural Network and Support Vector Machine applied to the field of text classification are described. The details of the approach are described in Gasanova *et al.* (2014b); Akhmedova *et al.* (2014). A metaheuristic nature-inspired algorithm proposed by Akhmedova & Semenkin (2013) is used as an optimization method.

4.2.2.1 Co-Operation of Biology Related Algorithms for Support Vector Machine

Support Vector Machine is one of the most popular classification methods, which has shown good performance on various text classification corpora. However, the choice of kernel function and parameters of SVM remains a complex task that requires expert knowledge.

In the most common formulation, support vector machines are classification mechanisms, which, given a training set

$$X^l = \{(x_1, y_1), \ldots, (x_l, y_l)\}, x_i \in R^m, y_i \in \{-1; 1\} \tag{4.17}$$

assuming l examples with m real attributes, learn a hyper-plane:

$$< w, x > +b = 0 \tag{4.18}$$

where $w \in R^m, b \in R, <, >$- dot product, which separates examples labelled as -1 from ones labelled as $+1$. Therefore, using this hyper-plane, a new instance x is classified using the following classifier:

$$\text{fitness} = \begin{cases} 1, (< w, x > +b) \geq 1 \\ -1 (< w, x > +b) \leq -1 \end{cases} \tag{4.19}$$

SVM is based on maximization of the distance between the discriminating hyper-plane

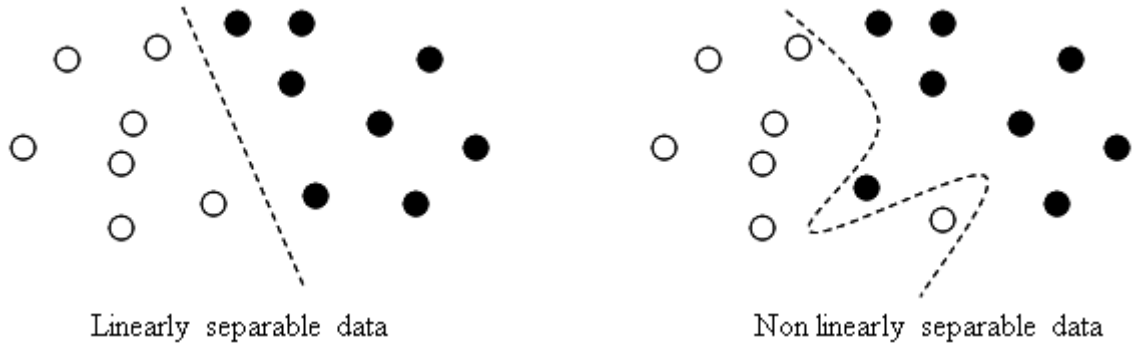

Figure 4.28: Linearly separable and non linearly separable data

and the closest examples. This maximization reduces the so-called structural risk, which is related to the quality of the decision function. The most discriminating hyper-plane can be computed by solving the following constrained optimization problem:

$$||w||^2 \to \min \tag{4.20}$$

$$y_i \left(< w, x_i > +b\right) \geq 1, i = \bar{1,l} \tag{4.21}$$

However, the given data set is not always linearly separable (Figure 4.28 provides an example of linearly separable and non separable data), and in this case SVM (as a linear classifier) does not provide satisfying classification results. One way to solve this problem is to map the data onto a higher dimension space and then to use a linear classifier in that space. The general idea is to map the original feature space to some higher-dimensional feature space where the training set is linearly separable. SVM provide an easy and efficient way of doing this mapping to a higher dimension space, which is referred to as the kernel trick.

In this study the polynomial kernel is used. So, let $K(x, x') = (\alpha < x, x' > +\beta)^d$ where α, β, d are parameters of the kernel function K. Then the classifier is:

$$\text{fitness} = \begin{cases} 1, (K(w,x) + \beta)^d + b \geq 1 \\ -1 \left(K(w,x) + \beta\right)^d + b \leq -1 \end{cases} \tag{4.22}$$

It means that the following constrained optimization problem should be solved:

$$||w||^2 \to \min \tag{4.23}$$

$$y_i \left((\alpha < w, x_i > +\beta)^d + b\right) \geq 1, i = \bar{1,l} \tag{4.24}$$

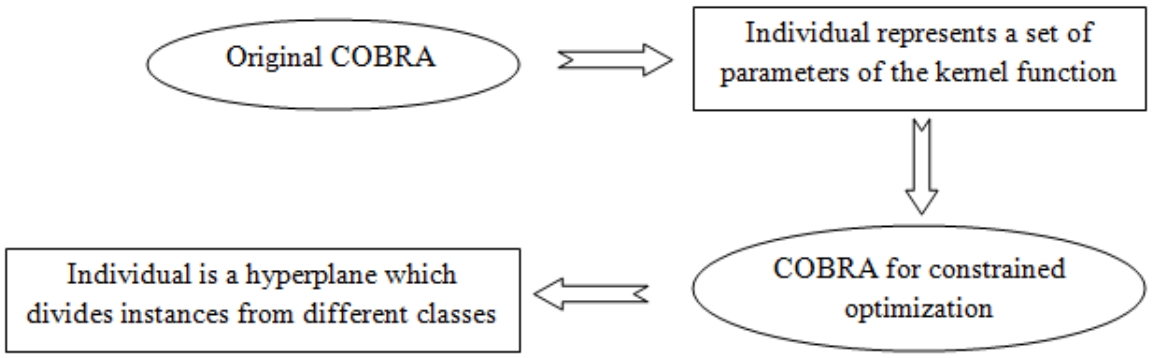

Figure 4.29: SVM generation and tuning using COBRA

Thus for solving a classification problem, kernel function's parameters , a vector w and a shift factor b should be determined, i.e. this constrained optimization problem with continuous variables has to be solved.

Thus for generating the SVM-machine the original COBRA is used: each individual in all populations represents a set of kernel function's parameters. Then for each individual constrained modification of COBRA is applied for finding vector w and the shift factor b. And finally, the individual with the best classification rate is chosen as the designed classifier. The details of this approach are presented in Akhmedova *et al.* (2014); Gasanova *et al.* (2014b).

The scheme of the structure generation and parameters tuning is presented in Figure 4.29.

4.2.2.2 Co-Operation of Biology Related Algorithms for Artificial Neural Network

It has been shown that in the number of text classification problems Artificial Neural Network (ANN) can produce good performance with carefully chosen structure and parameters. However, which structure and parameters should be taken to solve a given problem remains a complex problem.

The artificial neural network (ANN) model has three primary components: the input data layer, the hidden layer(s) and the output layer(s). Each of these layers contains nodes, and these nodes are connected to nodes at adjacent layer(s). The hidden layer includes two processes: the weighted summation function and the transformation function. Both of these functions relate the values from the input data to the output measures. Thereby "ANN's structure" consists of number of hidden layers, number of nodes (neurons) on each layer, and type of activation function on the node. Nodes in network are interconnected and each connection has a weight coefficient; the number of these coefficients depends on the solving problem (the number of inputs) and the number of hidden layers and nodes. Thus, networks with a more or less complex structure usually have too many weight coefficients which should

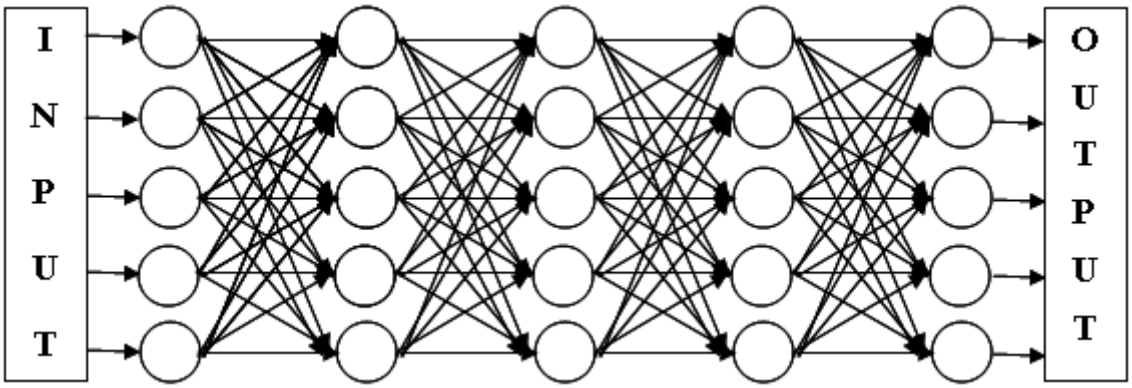

Figure 4.30: ANN model

be adjusted.

The weighted summation function is typically used in a feed forward/back propagation neural network model. However, there exist other optimization methods for training neural networks, which also show good results (e.g. in Sasaki & Tokoro (1999)).

Choosing the structure of the artificial neural network (ANN) and adjusting its weight coefficients are considered to be two unconstrained optimization problems: the first one with binary variables and the second one with real-valued variables. Type of variables depends on the representation of ANN's structure and coefficients. The details of this approach are presented in Akhmedova & Semenkin (2014).

First, let the maximum number of hidden layers be equal to 5 and the maximum number of neurons on each hidden layer be equal to 5, therefore, the maximum number of neurons in all layers is equal to 25. A larger number of layers and nodes can be chosen, but the aim of this work is to show that even the network with a relatively simple structure is able to produce good results if it is tuned with effective optimization techniques. Figure 4.30 illustrates the ANN model used in this thesis.

Each node is represented by the binary code of the length 4. If the code consists of zeros ("0000"), then this node does not exist in the given ANN. The whole structure of the neural network is represented by the binary code of the length 100 (25×4), each 20 variables represent one hidden layer. The number of input layers depends on the given problem. ANN has one output layer.

The following activation functions have been used for nodes:

1. $f(x) = 1/(1 + \exp(-x))$;
2. $f(x) = 1$;
3. $f(x) = \tanh(x)$;

4. $f(x) = \exp\left(-x^2/2\right)$;
5. $f(x) = 1 - \exp\left(-x^2/2\right)$;
6. $f(x) = x^2$;
7. $f(x) = x^3$;
8. $f(x) = \sin(x)$;
9. $f(x) = \exp(x)$;
10. $f(x) = |x|$;
11. $f(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \leq x \leq 1 \\ 1, & x > 1 \end{cases}$;
12. $f(x) = \begin{cases} 0, & x < -0.5 \\ x + 0.5, & -0.5 \leq x \leq 0.5 \\ 1, & x > 0.5 \end{cases}$;
13. $f(x) = 2/(1 + \exp(x)) - 1$;
14. $f(x) = 1/x$;
15. $f(x) = sign(x)$.

In order to determine which activation function will be used in the given node an integer number that corresponds to its binary code is calculated. For example, if the binary code "0101", then the integer number is equal to $0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 10$ and for this neuron the tenth activation function is used.

In order to find the best structure of ANN (the activation function for each node) the binary modification of COBRA is applied and the original version of COBRA that is designed for real-valued variables is used to adjust the weight coefficients for every structure. The scheme of the structure generation and parameters tuning is presented in Figure 4.31.

4.3 Summary

This chapter focuses on the development of novel text preprocessing methods, including a novel term weighting technique and a novel semi-supervised approach for feature space reduction that combines term clustering and weights recalculation of the obtained synthetic terms.

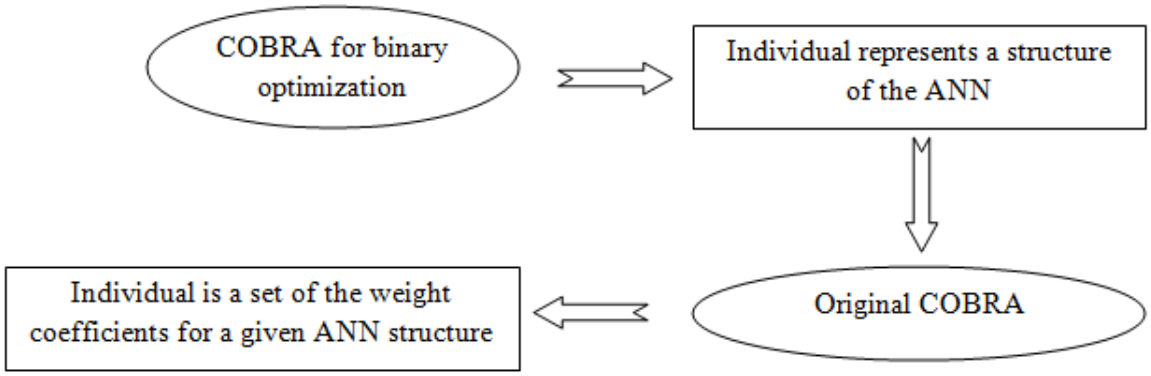

Figure 4.31: ANN generation and tuning using COBRA

First, a novel term weighting algorithm has been described. It is a supervised term weighting method based on the fuzzy rules membership formula. The main idea of term weighting algorithms is that each term has to contribute some value into the decision making process. In the proposed technique each term is assigned to the class, where statistically it appears the most. This algorithm calculates the weights for each word in the way that words that appeared significantly more in one class than in others have greater weights. These term weights contribute only to the corresponding class and the class with the greatest sum is assigned to the whole document. The method obtains similar results to the ConfWeight, however, it does not need the calculation of the t-distribution (Student's law), confidence intervals and other statistical functions required for the ConfWeight, and, therefore, the novel TW does not consume so much of CPU computational time.

One of the most important problems in text preprocessing is feature space reduction, since the term set obtained from the training data is often too large to directly use each word as a coordinate. The most commonly used methods to reduce the size of the term set involve the filtering according to the term weight, i.e. the subset of terms that weight less than a given value is removed from the feature space. These methods differ in the chosen term weighting function and the filtering margin. The advantage of such methods is its simplicity and speed, however, it throws out the information that can have a significant impact on the classification process. The filtering approach falls in the category of dimensionality reduction by feature selection. Another approach is based on the transformation of the initial features into the different feature space with the smaller dimension. Term clustering, latent semantic analysis, latent Dirichlet allocation, principal component analysis belong to the feature transformation approach to dimensionality reduction.

This chapter introduces a novel technique for reduction of the feature space dimension by clustering the terms with hierarchical agglomerative clustering. This algorithm has been applied for the corpora with large vocabulary and grammatically correct sentences, each of

the documents has many terms. In this work corpora from the DEFT 2007/2008 satisfy these criteria and SpeechCycle corpus has its own specialities which make the application of this approach pointless. The main reason is that each document (call) consists of only a few words which make great impact on the choice of the class. This clustering detect synonyms or term that have similar frequency distribution on categories. This algorithm can be applied with anyt term weighting techniques. After the initial terms are transformed into term clusters, the common weights for all term clusters are counted and the new data representation is recalculated.

Generally, after the dimensionality reduction of the feature space there is a significant loss of information and classification quality often decreases. However, since the size of the term set now is small, it is possible to apply an optimization method to adjust the term weights. This chapter proposes a novel weights recalculation method that improves the performance of classification algorithms. The optimization procedure is applied to the weights of the obtained term clusters since there is no evidence that the common weights of the term clusters lead even to the local optimum of the objective function. Since there is no information about the behaviour of the objective function heuristic approach should be considered. The cooperative scheme of coevolutionary genetic algorithm has been proposed as an optimization tool since the task to find optimal weights of the obtained term clusters can be naturally separated into the subtasks (subpopulations in the coevolutionary algorithm). Each subpopulation is considered as a category of the classification task, it evolves during several generations without sharing the information between subpopulations, then the evolution stops and algorithms updates the best obtained weights by collecting information from all subpopulations, then the cycle goes on. In this model individuals do not migrate from one subpopulation to another one, moreover, the individual from one subpopulation cannot be compared to the individual from another subpopulation. In order to provide the local convergence the local search algorithm is applied only to the best obtained individual on each generation. Local search algorithm inverts the random bit of the chromosome several times, fitness function is recalculated and if the change makes profit than it is kept.

Furthermore, since there is an overfitting problem on some of the corpora (Books and Games) that all applied optimization methods meet, the fitness function of the evolutionary algorithm (the optimization criterion) should be modified. Overfitting is a problem in optimization process, when the objective function reaches high values, however, the classification performance obtained on the previously unseen data is not good enough. In this chapter a semi-supervised modification of the fitness function has been introduced, which allows to improve the accuracy on the test set after the accuracy on the training set reaches almost 100%. The idea is to combine the existing supervised criterion with the unsupervised learning in order to obtain better partition into classes in terms of cluster analysis. In this work the

maximization of the accuracy rate obtained on the training set has been used as a supervised criterion and the maximization of the inter-cluster similarity (the average distance between centroids of the obtained clusters) has been used as an additional unsupervised criterion. This unsupervised criterion is added only to the candidate solutions which has already reached over 0.95 of accuracy rate on the training set. This change of the objective function leads to the improvement of the performance of the classification algorithms using this document representation, because of the better quality of the text preprocessing, since the cluster partition is more separate keeping the same accuracy rate on the training set.

Many real-world text classification problems include a category with different types of items, e.g. "non-uniform category", which consists of documents non-relevant to the given task. The detection of such non-uniform class is difficult for most of the classification algorithms. This chapter introduces a semi-supervised classification algorithm that improves the performance of classification methods in the corpora with non-uniform classes, which contain elements from different sources (such as SpeechCycle corpus with the class TE_NOMATCH). This method based on the assumption that if such non-uniform class is divided into the set of well-defined subclasses using an unsupervised learning algorithm, then the performance of the classification method will be improved. In this chapter the simulation results (obtained on SpeechCycle corpus) using genetic algorithm with integers, learning vector quantization algorithm trained with genetic algorithm and hierarchical agglomerative clustering with Hamming distance and different linkage criteria including single linkage, complete linkage, average linkage and Ward's criterion as clustering methods and the decision rule 4.3 as a classification algorithm have been provided and discussed. Numerical experiments have shown that the best performance has been obtained with the hierarchical agglomerative clustering algorithm (Ward's criterion).

All methods described in this chapter do not use any information specific for the domain or language, i.e. no stop word or ignore word lists have been used to filter the term set.

The performance of the proposed text preprocessing algorithms compared with the results of the state-of-the-art text preprocessing techniques is presented in Chapter 6. The algorithms settings used in the simulations and implementation details are also shown in Chapter 6.

In order to estimate the quality of the obtained text representation, one has to apply several classification algorithms. It is important to check the performance on various classification algorithms, since some preprocessing techniques work better only with the special classifier.

In the field of text classification the most popular methods are Support Vector Machine (SVM) and Artificial Neural Network (ANN), since they produce better performance on various domains with different text representations. However, in order to produce such performance they require a careful choice of parameters that is usually made by experts. In the this chapter the novel approach modified for generation and parameters adjustment of SVM- and

ANN-based classifiers is introduced.

This chapter describes contributions to the field of text classification. It consists of two main parts. The first one is related to the classification performance improvement achieved by combining supervised and unsupervised learning models. The second part is concerned with optimization of a structure and parameters of Support Vector Machine and Artificial Neural Network applied for text classification task.

In this chapter an optimization of a metaheuristic method called Co-Operation of Biology Related Algorithms (COBRA) and its modifications for the text classification task has been described. This metaheuristic approach combines five different heuristic algorithms: Particle Swarm Optimization (PSO) proposed in Kennedy *et al.* (1995), Wolf Pack Search (WPS) proposed in Yang *et al.* (2007), Firefly Algorithm (FFA) proposed in Yang (2009), Cuckoo Search Algorithm (CSA) proposed in Yang & Deb (2009) and Bat Algorithm (BA) proposed in Yang (2010). The original version of COBRA has been designed for unconstrained optimization in real-valued vector space, a constrained version uses Deb's rule, dynamic penalties (both for WPS, FFA, CSA and BA) and the technique proposed in Liang *et al.* (2010) (for PSO) to modify the component algorithms. For a binary modification the value of sigmoid function from the position of particles is calculated and this value is considered as a probability that the corresponding position equals to 0.

This chapter presents an implementation of the original version of COBRA, its constrained modification and its binary modification have been applied for automated design and tuning parameters of Support Vector Machine (SVM) and Artificial Neural Network (ANN) text classifiers. The source of SVM effectiveness in comparison with other SVM algorithms is the additional flexibility of the separating surfaces generated by the implemented approach. In order to choose the best structure (the activation function for each node) binary COBRA has been applied and to adjust the weight coefficients for each structure the original version has been used. This ANN model and metaheuristic have been chosen, since they produce relatively small and flexible enough neural network, which is able to outperform other more complex systems.

In the next chapter the text classification algorithms described in this chapter are compared with the state-of-the-art approaches using different text preprocessing methods. The algorithms settings used in the simulations and implementation details are also shown in the following chapter.

Chapter 5

Implementation and Evaluation

5.1 Implementation

This chapter presents an experimental setup used for the numerical experiments conducted in this thesis: applied classification algorithms and text preprocessing methods. It also includes simulation results in terms of classification quality (F-score and accuracy) and CPU computational time required for different text preprocessing methods obtained on the corpora. The comparison of the results obtained on the initial feature space, on the feature space reduced using term filtering and the feature space reduced using the semi-supervised term clustering algorithm.

5.1.1 Classification Methods

Numerical experiments have been conducted using several popular text classification algorithms. The methods have been implemented using RapidMiner (Shafait *et al.* (2010)) and Microsoft Visual Studio C++ 2010. The applied classification algorithms include:

- k-nearest neighbors algorithm with weighted vote (RapidMiner);
  - Weighted vote;
    If this parameter is set, the weight of examples is also taken into account. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more than the more distant ones.
  - Number of neighbors: k varies from 1 to 15;
  - Measure types: MixedMeasures.
    - Mixed measure: MixedEuclideanDistance
- kernel Bayes classifier with Laplace correction (RapidMiner);

- Laplace correction;
  This parameter indicates if Laplace correction should be used to prevent high influence of zero probabilities. There is a simple trick to avoid zero probabilities. It can be assumed that the training set is so large that adding one to each count would only make a negligible difference in the estimated probabilities, yet would avoid the case of zero probability values. This technique is known as Laplace correction.
- Estimation mode: greedy;
  This parameter specifies the kernel density estimation mode. Two options are available.
  - full: If this option is selected, you can select a bandwidth through heuristic or a fix bandwidth can be specified.
  - greedy: If this option is selected, you have to specify the minimum bandwidth and the number of kernels.
- Minimum bandwidth: 0,1;
  This parameter is only available when the estimation mode parameter is set to 'greedy'. This parameter specifies the minimum kernel bandwidth.
- number of kernels : 10.
  This parameter is only available when the estimation mode parameter is set to 'greedy'. This parameter specifies the number of kernels.

- Artificial Neural Network with error back propagation (RapidMiner);
  This operator learns a model by means of a feed-forward neural network trained by a back propagation algorithm (multi-layer perceptron).
  - Training cycles: 500;
    This parameter specifies the number of training cycles used for the neural network training. In back-propagation the output values are compared with the correct answer to compute the value of some predefined error-function. The error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. This process is repeated n number of times.
  - Learning rate: 0,3;
    This parameter determines how much the weights are changed at each step. It should not be 0.
  - Momentum: 0,2;
    The momentum simply adds a fraction of the previous weight update to the current

one. This prevents local maxima and smoothes optimization directions.

- Error epsilon: 10-5;
  The optimization is stopped if the training error gets below this epsilon value.
- Shuffle.
  This is an expert parameter. It indicates if the input data should be shuffled before learning. Although it increases memory usage but it is recommended if data is sorted before) and normalizing (This is an expert parameter. The Neural Net operator uses an usual sigmoid function as the activation function. Therefore, the value range of the attributes should be scaled to -1 and +1. This can be done through the normalize parameter. Normalization is performed before learning. Although it increases runtime but it is necessary in most cases.

- Artificial Neural Network (ANN) generated and adjusted with Co-Operation of Biology Related Algorithms (COBRA) (Microsoft Visual Studio 2010);
  The problem of ANN-based classifier generation can be divided into finding the structure and tuning the weight coefficients. The optimization method for problems with binary variables (binary COBRA) for finding the best structure is used and the optimization method for problems with real-valued variables (original COBRA) for every structure weight coefficients adjustment is applied.

- Rocchio Classifier with different metrics and γ parameter (Microsoft Visual Studio 2010);
  Rocchio classifier (Rocchio (1971)) is a well-known classifier based on the search of the nearest centroid. For each category a weighted centroid is calculated.

  - γ is parameter corresponds to relative importance of negative precedents. The given document is put to the class with the nearest centroid. In this thesis the Rocchio classifier has been applied with $\gamma \in (0.1; 0.9)$ and with three different metrics: taxicab distance, Euclidean metric and cosine similarity.

- Support Vector Machine (SVM) with linear kernel and ML (RapidMiner);

  - Fast large margin;
    The Fast Large Margin operator applies a fast margin learner based on the linear support vector learning scheme proposed by R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. Although the result is similar to those delivered by classical SVM or logistic regression implementations, this linear classifier is able to work on data set with millions of examples and attributes.
  - Solver: L2 SVM Dual;
    This parameter specifies the solver type for this fast margin method.

  - C: 1.0;
    This parameter specifies the cost parameter C. It is the penalty parameter of the error term.
  - Epsilon: 0,01;
    This parameter specifies the tolerance of the termination criterion.
  - use bias.
    This parameter indicates if an intercept value should be calculated.

- Support Vector Machine (SVM) generated and optimized with Co-Operation of Biology Related Algorithms (COBRA) (Microsoft Visual Studio 2010).
  For generating the SVM-machine the original COBRA is used: each individual in all populations represents a set of kernel function's parameters. Then for each individual constrained modification of COBRA is applied for finding vector w and the shift factor b. And finally, the individual with the best classification rate is chosen as the designed classifier.

5.1.2 Text Preprocessing Methods

Text preprocessing algorithms are based on the assumption that the class of the document depends on the words in this document. The simplest approach is a binary representation, where each word of the document is considered as a binary coordinate and the dimensionality of the feature space is the number of words appeared in the collection. The problems of this approach are related with the high dimensionality, since the great number of words leads to the difficulties for most classification methods to perform well.

In order to overcome this problem text preprocessing usually includes feature extraction, term weighting and dimensionality reduction. In the stage of feature extraction what will be considered as features or terms is decided. In this thesis single words are taken as terms after removal of punctuation signs and transforming all letters into lower case. However, no other information related to domain or language is not used. Most of text classification systems that have been for results comparison (participants of DEFT 2007/2008) apply at least stop-word filtering, which removes function words and other linguistic or domain related data.

In the stage of term weighting the "importance" of the term in the given document is measured based on the statistical information about term occurrence in the training set.

In the numerical experiments, nevertheless, a binary representation is used in comparison with 10 other text preprocessing methods including 8 unsupervised term weighting techniques (TF-IDF modifications), an existing supervised term weighting called ConfWeights and a novel supervised approach introduced in this thesis (novel TW). For detailed description of TF-IDF modifications and ConfWeight see Chapter 2.1 and for the novel TW - Chapter 4.1.2.

All modifications of TF-IDF are calculated using the formula $TF * IDF_{ij} = tf_{ij} * idf_i$, where $tf_{ij} = \frac{t_i}{\sum_{l=1}^{T} t_l}$, t_i is the number of times the ith word occurs in the jth document; T is the document size. TF-IDF 3 and TF-IDF 4 are used with 3 different values of the parameter $\alpha = 0.1, 0.5, 0.9$ (there are three document representations with TF-IDF 3 and three representations with TF-IDF 4).

ConfWeight uses another variant of tf_{ij}: $ConfWeight_{ij} = \log(tf_{ij}+1) \cdot Maxstr(i)$. Novel TW approach is calculated with t_i instead of tf_{ij}.

Table 5.1 shows the term weighting methods implemented in this thesis.

5.2 Algorithmic Comparison of the Preprocessing Methods

5.2.1 Opinion Mining Corpora

For the opinion mining task three publically available corpora from the DEFT 2007 campaign ("Défi Fouille de Texte") have been used for algorithms application and results comparison. The corpora are: Books, Games and Debates (for detailed information see Chapter 3.1).

In order to directly compare the obtained performance with the results of participants the same measures of classification quality have to be used: precision, recall and F-score . Classification methods applied on these corpora include

- k-nearest neighbors method with weighted vote (k varies from 1 to 15);
- kernel Bayes classifier with Laplace correction;
- linear Support Vector Machine (SVM);
- Support Vector Machine with ML;
- Support Vector Machine optimized using COBRA;
- Rocchio classifier;
- Artificial Neural Network (ANN) with error back propagation learning;
- Artificial Neural Network (ANN) optimized using COBRA.

These corpora are divided into the training (60%) and the test (40%) sets by the organizers of the campaign and this partition has been kept to be able to directly compare the performance achieved using the methods developed in this thesis with the algorithms of participants.

Tables 5.2, 5.3 and 5.4 present the F-scores obtained on the test corpora. The best values are shown in bold. Results of the k-NN algorithm are presented with the best k parameter.

Table 5.2 shows that the best classification quality ($Fscore = 0.619$) is provided with the novel TW approach as text preprocessing and Support Vector Machine generated using

Preprocessing	Term Weighting
TF-IDF 1	$idf_i = \log \frac{\|D\|}{n_i}$, where n_i is the number of documents which have the ith word. $\|D\|$ is the number of document in the training set.
TF-IDF 2	$idf_i = \log \frac{\|D\|}{n_i}$, where n_i is the number of times ith word appears in all documents from the training set. $\|D\|$ is the number of document in the training set.
TF-IDF 3	$idf_i = \left(\frac{\|D\|}{n_i}\right)^{\alpha}, \alpha \in (0,1)$, where n_i is the number of documents which have the ith word; α is the parameter (3 variants: $\alpha = 0.1, 0.5, 0.9$) $\|D\|$ is the number of document in the training set.
TF-IDF 4	$idf_i = \left(\frac{\|D\|}{n_i}\right)^{\alpha}, \alpha \in (0,1)$, where n_i is the number of times ith word appears in all documents from the training set; α is the parameter (3 variants: $\alpha = 0.1, 0.5, 0.9$) $\|D\|$ is the number of document in the training set.
ConfWeight	$p(x,n) = \frac{x + 0.5z^2_{\alpha/2}}{n + z^2_{\alpha/2}}$, where $\Phi(z_{\alpha/2}) = \frac{\alpha}{2}$, $\Phi$is Student law; in this work $\alpha = 0.95$ and $0.5z^2_{\alpha/2} = 1.96$ (as recommended by the authors of ConfWeight); for each feature f and each class c two functions are calculated: $p_+(x,n) = p(x,n)$ where x is number of vectors which belong to the class c and have non-zero feature f; n is the number of documents which belong to the class c. $p_-(x,n) = p(x,n)$ where x is number of vectors which have the feature f but do not belong to the class c; n is the number of documents which do not belong to the class c. Interval $(\underline{p}, \overline{p})$ where $\underline{p} = p - 0.5z^2_{\alpha/2}\sqrt{\frac{p(1-p)}{n + z^2_{\alpha/2}}}$, $\overline{p} = p + 0.5z^2_{\alpha/2}\sqrt{\frac{p(1-p)}{n + z^2_{\alpha/2}}}$ $str_{f,c} = \begin{cases} if\ \left(\underline{p_+} > \overline{p_-}\right)\ than\ str_{f,c} = \log_2(2\frac{\underline{p_+}}{\underline{p_+} + \overline{p_-}}), \\ else\ str_{f,c} = 0 \end{cases}$ $Maxstr(f) = (\max_{c \in C} str_{f,c})^2$
Novel TW	$C_j = \frac{1}{\sum_{i=1}^{L} T_{ji}}(R_j - \frac{1}{L-1}\sum_{i=1\, i \neq S_j}^{L} T_{ji})$, where L is the number of classes; n_i is the number of documents which belong to the ith class; N_{ji} is the number of times jth term occur in the documents of the ith class; $T_{ji} = \frac{N_{ji}}{n_i}$ is the relative frequency of jth term in the ith class; $R_j = \max_i T_{ji}$; $S_j = \arg(\max_i T_{ji})$ is the number of class "assigned" to jth term.

Table 5.1: Term weighting methods applied in this thesis

	k-**NN**	**Bayes**	**SVM** Linear	**SVM** FLM	**SVM** CO-BRA	**Roc-chio**	**ANN**	**ANN** CO-BRA
Novel TW	0,488 (14)	0,437	**0,516**	0,486	**0,619**	0,537	0,493	0,585
Binary	0,488 (12)	0,489	0,498	0,509	0,558	0,479	0,475	0,566
Conf Weight	**0,559 (15)**	0,238	0,238	**0,534**	0,588	**0,557**	**0,570**	**0,613**
tf.idf 1	0,517 (3)	0,495	0,499	0,500	0,580	0,488	0,505	0,554
tf.idf 2	0,516 (1)	**0,506**	0,511	0,495	0,564	0,498	0,505	0,533
tf.idf 3 $\gamma = 0.1$	0,494 (7)	0,489	0,487	0,486	0,574	0,489	0,489	0,518
tf.idf 3 $\gamma = 0.5$	0,473 (3)	0,478	0,471	0,486	0,577	0,462	0,472	0,460
tf.idf 3 $\gamma = 0.9$	0,449 (1)	0,451	0,441	0,463	0,558	0,427	0,437	0,505
tf.idf 4 $\gamma = 0.1$	0,502 (8)	0,484	0,490	0,424	0,550	0,490	0,498	0,554
tf.idf 3 $\gamma = 0.5$	0,473 (4)	0,490	0,476	0,487	0,559	0,464	0,471	0,551
tf.idf 4 $\gamma = 0.9$	0,443 (1)	0,465	0,437	0,460	0,561	0,420	0,439	0,542

Table 5.2: Classification results obtained on the Books corpus

COBRA as a classification method. This result outperforms the one obtained by the DEFT'07 participants even so no term filtering has been used in text preprocessing (the best $Fscore = 0.603$ and the average $Fscore = 0.5004$). The second result is obtained using the ConfWeight preprocessing and k-NN ($k = 15$), Rocchio or Neural Network as classification methods. However, it should be mentioned that in general the performance of the SVM (COBRA) and the ANN (COBRA) is better than any other methods.

Table 5.3 presents the classification results obtained on the test set of the Games corpus. The best classification quality ($Fscore = 0.727$) is provided with the ConfWeight as text preprocessing and ANN generated and trained using COBRA as a classification method. The results provided by ConfWeight with Rocchio Classifier, ConfWeight with k-NN ($k = 15$) and the novel TW with Rocchio are close to the best ($Fscore = 0.717$, $Fscore = 0.720$ and $Fscore = 0.712$) but novel TW is computed in about three times faster. These results are better than the average performance obtained by DEFT'07 participants (the best $Fscore = 0.78$ and the average $Fscore = 0.6604$). However, the systems developed by the participants take

	k-NN	Bayes	SVM RM	SVM COBRA	Rocchio	ANN RM	ANN COBRA
Novel TW	0,699 (13)	**0,675**	0,675	**0,696**	0,712	0,691	0,692
Binary	0,703 (5)	0,653	0,668	0,682	0,659	0,707	0,654
Conf Weight	**0,720 (15)**	0,210	0,210	0,645	**0,717**	**0,717**	0,**727**
tf.idf 1	0,672 (3)	0,652	0,665	0,669	0,659	0,677	0,642
tf.idf 2	0,671 (7)	0,651	0,661	0,661	0,654	0,664	0,649
tf.idf 3 $\gamma = 0.1$	0,691 (8)	0,628	**0,685**	0,687	0,678	0,679	0,679
tf.idf 3 $\gamma = 0.5$	0,642 (3)	0,633	0,640	0,660	0,637	0,647	0,644
tf.idf 3 $\gamma = 0.9$	0,602 (1)	0,597	0,600	0,604	0,606	0,606	0,628
tf.idf 4 $\gamma = 0.1$	0,701 (12)	0,623	0,682	0,691	0,681	0,678	0,701
tf.idf 3 $\gamma = 0.5$	0,655 (7)	0,643	0,647	0,645	0,637	0,657	0,679
tf.idf 4 $\gamma = 0.9$	0,588 (13)	0,582	0,576	0,657	0,593	0,575	0,552

Table 5.3: Classification results obtained on the Games corpus

advantage of the linguistic information and domain-related information, since their algorithms have been carefully chosen for these corpora.

The performance of the developed algorithms compared with the existing state-of-the-art methods on the opinion mining corpus Debates is shown in Table 5.4. The best classification quality ($Fscore = 0.714$) is provided with the ConfWeight as text preprocessing and SVM generated and tuned using COBRA as a classification method. This result is better than the average performance and close to the best performance obtained by DEFT'07 participants (The best result obtained by Torres-Moreno: $Fscore = 0.720$ and the average $Fscore = 0.642$). However, the results provided by ConfWeight in combination with Artificial Neural Networks (RapidMiner and COBRA) and the novel TW in combination with SVM (RapidMiner) are close to the best ($Fscore = 0.705; 0.709$ and $Fscore = 0.702$), but the novel TW is computed in about three times faster.

Tables 5.2, 4.3 and 5.4 show that the best F-scores have been obtained mostly with either ConfWeight or the novel TW preprocessing method. The algorithm performances on the Games and Debates corpora achieved the best results with ConfWeight; however, the F-scores

	k-**NN**	**Bayes**	**SVM** RM	**SVM** CO-BRA	**Roc-chio**	**ANN** RM	**ANN** CO-BRA
Novel TW	0,695 (15)	0,616	**0,702**	0,700	0,696	0,697	0,705
Binary	0,645 (15)	0,555	0,655	0,642	0,636	0,654	0,647
Conf Weight	**0,695 (15)**	0,363	0,634	**0,714**	**0,697**	**0,705**	**0,709**
tf.idf 1	0,637 (15)	0,637	0,642	0,638	0,638	0,638	0,640
tf.idf 2	0,634 (15)	0,639	0,640	0,641	0,638	0,632	0,641
tf.idf 3 $\gamma = 0.1$	0,645 (9)	0,644	0,651	0,661	0,630	0,645	0,646
tf.idf 3 $\gamma = 0.5$	0,607 (15)	0,608	0,609	0,669	0,606	0,611	0,612
tf.idf 3 $\gamma = 0.9$	0,570 (14)	0,561	0,565	0,669	0,562	0,566	0,644
tf.idf 4 $\gamma = 0.1$	0,648 (13)	**0,645**	0,650	0,663	0,646	0,647	0,643
tf.idf 3 $\gamma = 0.5$	0,607 (13)	0,611	0,609	0,665	0,603	0,610	0,601
tf.idf 4 $\gamma = 0.9$	0,570 (15)	0,560	0,564	0,661	0,558	0,565	0,566

Table 5.4: Classification results obtained on the Debates corpus

obtained with the novel TW preprocessing are very similar (0.712 and 0.720 for Games; 0.700 and 0.714 for Debates). All best results have been obtained with SVM or ANN generated and optimized using COBRA, which shows the effectiveness of the metaheuristic methods applied for the classifier generation. The novel TW is more effective in comparison with all modifications of TF-IDF combined with all classification methods. Binary preprocessing has in average the worst performance than TF-IDF with all classification methods.

This thesis focuses on the text preprocessing methods that do not require language or domain-related information; therefore, the aim was not to achieve the best possible classification quality but to develop fully statistical text classification algorithms that are able to compete with the existing systems designed for the text classification tasks that use stop-word list and other a-priori known information. However, the result obtained on Books corpus with the novel TW preprocessing technique and SVM (generated using COBRA) as a classification algorithm has reached 0.619 F-score and the result with the novel TW preprocessing algorithm and ANN (generated using COBRA) has reached 0.613 F-score, which are both higher

than the best known performance 0.603 (Proceedings of the 3rd DEFT Workshop, 2007). Performances on other corpora have achieved close F-score values to the best submissions of the DEFT'07 participants.

5.2.2 Topic Categorization

5.2.2.1 Document Classification

The DEFT ("Défi Fouille de Texte") Evaluation Package 2008 provided by ELRA has been used for algorithms application and results comparison. In order to evaluate obtained results with the campaign participants the same measures of classification quality have to be applied: precision, recall and F-score. Implemented classification methods are:

- k-nearest neighbors method with weighted vote (k varies from 1 to 15);
- kernel Bayes classifier with Laplace correction;
- linear Support Vector Machine (SVM);
- Support Vector Machine with ML;
- Support Vector Machine generated using COBRA;
- Rocchio classifier;
- decision rule 4.3;
- Artificial Neural Network (ANN) with error back propagation learning;
- Artificial Neural Network (ANN) generated using COBRA.

These corpora are divided into the training (60%) and the test (40%) sets by the organizers of the campaign and this partition has been kept to be able to directly compare the performance achieved using the methods developed in this thesis with the algorithms of participants.

Tables 5.5 and 5.6 present the F-scores obtained on the test corpora. The best values are shown in bold. Results of the k-NN algorithm are presented with the best k parameter.

Table 5.5 presents the classification results obtained on the corpus T1. The best classification quality ($Fscore = 0.876$) is provided with the ConfWeight as a text preprocessing algorithm and SVM generated using COBRA as a classification method. However, the result provided by the novel TW in combination with the SVM text classifier trained using COBRA is close to the best one ($Fscore = 0.867$) but the novel TW is computed in about four times faster. These result are better than the average performance obtained by DEFT'08 participants (the best: $Fscore = 0.894$ and the average $Fscore = 0.826$).

	k-**NN**	**Bayes**	**SVM** Linear	**SVM** ML	**SVM** CO-BRA	**Rocchio**	**Dec. rule 4.3**	**ANN** RM	**ANN** CO-BRA
Novel TW	0,837 (13)	0,794	0,834	**0,856**	0,867	0,849	**0,838**	**0,854**	0,614
Binary	0,800 (11)	0,501	0,775	0,788	0,788	0,794	0,728	0,783	0,579
Conf Weight	**0,855 (14)**	**0,837**	**0,848**	0,840	**0,876**	**0,853**	0,832	0,853	0,625
tf.idf 1	0,816 (15)	0,591	0,804	0,825	0,838	0,825	0,807	0,830	**0,644**
tf.idf 2	0,808 (15)	0,690	0,812	0,827	0,828	0,817	0,803	0,808	0,513
tf.idf 3 $\gamma = 0.1$	0,816 (10)	0,555	0,811	0,819	0,823	0,824	0,781	0,822	0,582
tf.idf 3 $\gamma = 0.5$	0,782 (5)	0,613	0,711	0,807	0,818	0,789	0,790	0,777	0,550
tf.idf 3 $\gamma = 0.9$	0,718 (15)	0,576	0,598	0,741	0,748	0,721	0,725	0,667	0,569
tf.idf 4 $\gamma = 0.1$	0,816 (15)	0,606	0,808	0,824	0,827	0,824	0,786	0,825	0,618
tf.idf 4 $\gamma = 0.5$	0,776 (7)	0,624	0,759	0,808	0,809	0,782	0,781	0,768	0,563
tf.idf 4 $\gamma = 0.9$	0,704 (9)	0,576	0,598	0,742	0,740	0,708	0,711	0,646	0,598

Table 5.5: Classification results obtained on the T1 corpus

	k-NN	Bayes	SVM Linear	SVM ML	Rocchio	Dec. rule 4.3	ANN RM
Novel TW	**0,811 (15)**	**0,745**	**0,852**	**0,851**	**0,834**	**0,803**	**0,843**
Binary	0,728 (13)	0,569	0,799	0,794	0,765	0,388	0,799
Conf Weight	0,785 (13)	0,712	0,815	0,813	0,803	0,771	0,820
tf.idf 1	0,786 (15)	0,658	0,830	0,837	0,825	0,778	0,838
tf.idf 2	0,775 (15)	0,728	0,830	0,830	0,823	0,780	0,826
tf.idf 3 $\gamma = 0.1$	0,761 (14)	0,591	0,823	0,823	0,797	0,575	0,817
tf.idf 3 $\gamma = 0.5$	0,759 (14)	0,691	0,808	0,818	0,812	0,796	0,801
tf.idf 3 $\gamma = 0.9$	0,677 (15)	0,663	0,563	0,740	0,741	0,739	0,716
tf.idf 4 $\gamma = 0.1$	0,765 (15)	0,592	0,824	0,824	0,799	0,606	0,807
tf.idf 3 $\gamma = 0.5$	0,754 (15)	0,695	0,802	0,814	0,809	0,795	0,796
tf.idf 4 $\gamma = 0.9$	0,664 (14)	0,648	0,660	0,724	0,726	0,723	0,690

Table 5.6: Classification resultsobtained on the T2 corpus

The performance of the classification algorithms obtained on the test set of the topic categorization task T2 is presented in Table 5.6. The best classification quality ($Fscore = 0.852$) is provided with the novel TW as a text preprocessing algorithm and the linear SVM as a classification method. This result is better than the average performance obtained by DEFT'08 participants (the best $Fscore = 0.880$ and the average $Fscore = 0.811$). It should be mentioned that on this corpus performance of all applied algorithms are better using the novel TW.

On the document classification corpora the best performance obtained in this thesis has been achieved using the novel TW to preprocess the textual data and SVM to classify the preprocessed documents.

	k-NN	Bayes	SVM linear	Artificial Neural Network
Binary	0.753 (8)	**0.720**	0.651	0.654
Conf Weight	0.800 (4)	0.489	**0.738**	**0.770**
Novel TW	**0.803 (4)**	0.713	0.695	0.710
tf.idf 1	0.673 (3)	0.674	0.585	0.637
tf.idf 2	0.656 (7)	0.676	0.585	0.637
tf.idf 3 $\gamma = 0.1$	0.693 (6)	0.663	0.626	0.643
tf.idf 3 $\gamma = 0.5$	0.612 (6)	0.665	0.489	0.574
tf.idf 3 $\gamma = 0.9$	0.581 (6)	0.684	0.404	0.362
tf.idf 4 $\gamma = 0.1$	0.693 (6)	0.663	0.626	0.643
tf.idf 4 $\gamma = 0.5$	0.612 (4)	0.665	0.489	0.574
tf.idf 4 $\gamma = 0.9$	0.580 (6)	0.684	0.404	0.362

Table 5.7: Classification results for SpeechCycle

5.2.2.2 Natural Language Call Routing

Natural language call routing database provided by the SpeechCycle company has been used for the algorithms application and the results comparison. In order to evaluate the algorithms performances and to compare results the accuracy rate has been used as a classification quality measure. Different classification algorithms have been implemented using RapidMiner (Shafait *et al.* (2010)) in order to compare the performance of the proposed text preprocessing algorithm with the state-of the-art techniques. The classification methods are:

- k-nearest neighbors method with weighted vote (k varies from 1 to 15);
- kernel Bayes classifier with Laplace correction;
- linear Support Vector Machine (SVM);
- Artificial Neural Network with error back propagation learning.

The measure of effectiveness is the classification accuracy obtained on the test set. The results are presented in Table 5.7. There are the best values of k for k-NN in brackets.

Table 5.7 shows that the best classification accuracy ($accuracy = 0.803$) is provided with the novel TW as a text preprocessing technique and k-nearest neighbors algorithm as a classification method ($k = 4$). The performance of the k-NN with the ConfWeight preprocessing is similar ($accuracy = 0.800$), however, ConfWeight requires more computational time than the novel TW. The classification accuracy of the novel TW is less than the accuracy of ConfWeight with SVM or neural network, but novel TW approach is more effective with Bayes algorithm. TW is more effective in comparison with all modifications of TF-IDF with all classification methods. Binary preprocessing is more effective than TF-IDF with all classification methods. This can be explained by the fact that the database contains very short calls (often only one word) and repeatability of words in one call is close to zero. TF-IDF is more appropriate for large documents with large number of repetitive words.

5.2.3 Time Comparison of the Preprocessing Methods

5.2.3.1 Opinion Mining and Document Classification

For Books, Games, Debates, T1 and T2 corpora 11 different text preprocessing methods (8 modifications of TF-IDF, binary method, ConfWeight and novel TW method) have been implemented. The computational efficiency of each text preprocessing technique has been evaluated. Each text preprocessing method has been run 20 times using the Baden-Württemberg Grid (bwGRiD) Cluster Ulm (Every blade comprehends two 4-Core Intel Harpertown CPUs with 2.83 GHz and 16 GByte RAM).

Figure 5.1 compares average computational times for different preprocessing methods applied on DEFT'07. The computational time of all TF-IDF modifications are represented by the average value because the time variation for the modifications is not significant. It varies from 12 to 17 minutes for Books corpus, from 13 to 14 minutes for Games corpus and from 28 to 30 minutes for Debates corpus.

Figure 5.1 shows that TF-IDF modifications and the novel TW require almost the same computational time. The most time-consuming method is ConfWeight (CW). It requires approximately two times more computational time than TF-IDF and the novel TW for DEFT'07 corpora.

Figure 5.2 compares average computational times for different preprocessing methods applied on DEFT 2008. For these corpora the average computational time for all TF-IDF preprocessing has been presented since there is no significant difference. It varies from 33 to 39 minutes for T1 corpus and from 70 to 79 minutes for T2 corpus.

Figure 5.2 shows that TF-IDF and the novel TW require almost the same computational time. The most time-consuming method is ConfWeight (CW). It requires approximately five times more computational time than TF-IDF and the novel TW for DEFT'08 corpora.

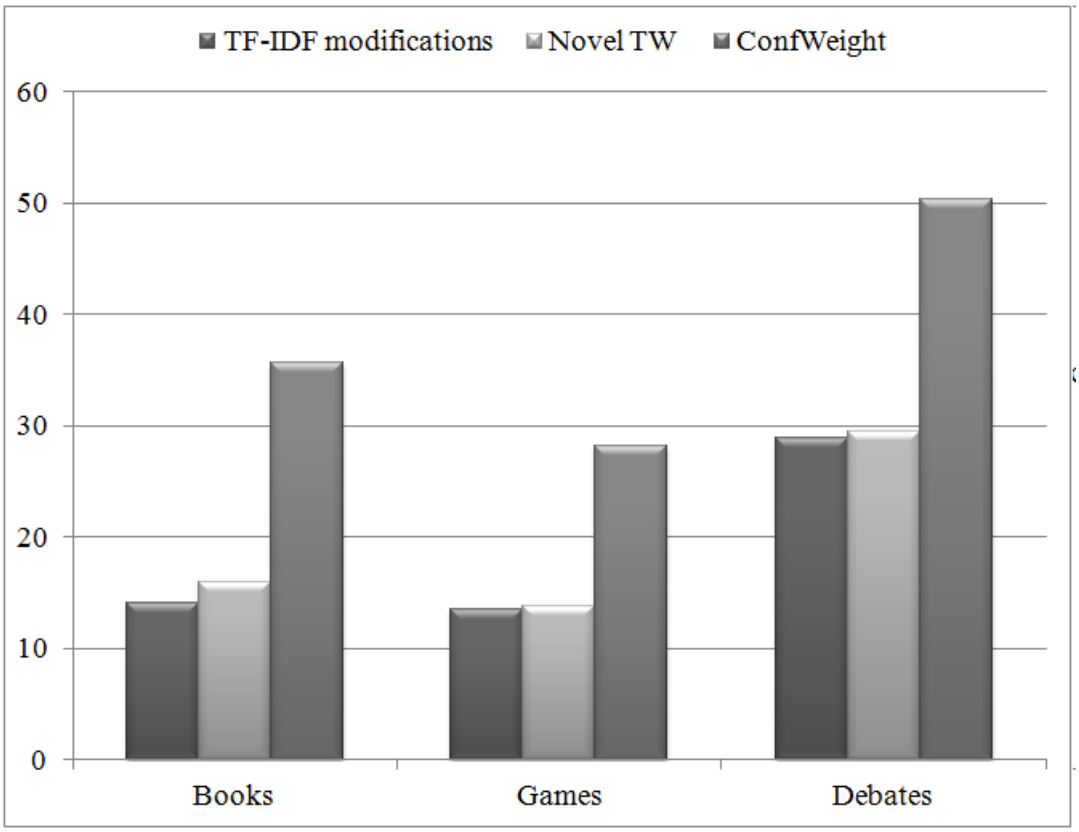

Figure 5.1: Computational time (min) obtained using different text preprocessing methods on the DEFT 2007 corpora (the TF-IDF modifications time is represented by the average values of all TF-IDF modifications)

5.2.3.2 Natural Language Call Routing

For SpeechCycle corpus 11 different text preprocessing methods (8 modifications of TF-IDF, binary method, ConfWeight and novel TW method) have been implemented. First, the computational efficiency of each text preprocessing technique has been evaluated. Each text preprocessing method has been run 20 times with the same computer (Intel Core i7 2.90 GHz, 8 GB RAM).

Figure 5.3 compares computational times required for different preprocessing methods. The average value for all TF-IDF modifications is presented in the Figure 5.3 because the time variation for the modifications is not significant (it varies from 22.2 min to 22.8 min).

Figure 5.3 shows that the binary preprocessing is the fastest one. TF-IDF modifications and the novel TW are approximately one and a half times slower than the binary preprocessing and they have almost the same computational efficiency. The most time-consuming method is ConfWeight (CW). It requires approximately eight times more time than TF-IDF and the novel TW.

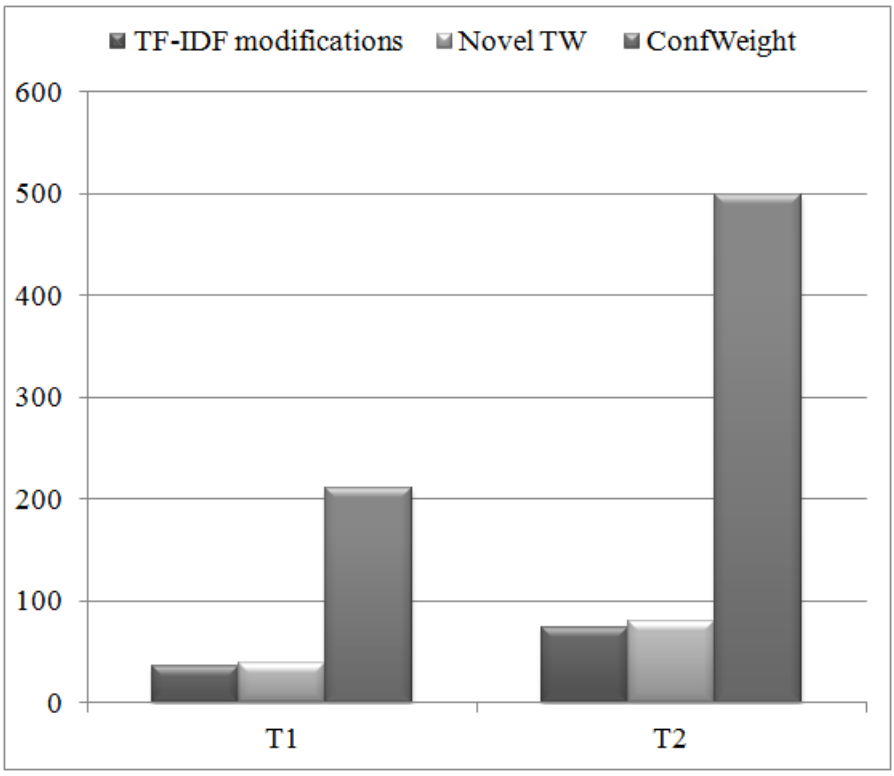

Figure 5.2: Computational time (min) obtained using different text preprocessing methods on the DEFT 2008 corpora (the TF-IDF modifications time is represented by the average values of all TF-IDF modifications)

5.3 Simulations and Results with Reduced Feature Space

In this section the numerical experiments conducted with the reduced term sets obtained using the proposed semi-supervised dimensionality reduction method. This algorithm has been applied only to the DEFT 2007/2008 corpora, since the feature space of SpeechCycle data is not high dimensional.

5.3.1 Reduced Feature Space using Feature Selection

Three publically available corpora for opinion mining (corpora: Books, Games and Debates) presented in def (2007) and the DEFT ("Défi Fouille de Texte") Evaluation Package 2008 (corpora: T1 and T2) provided by ELRA have been used for algorithms application and results comparison. In order to evaluate the obtained results precision, recall and F-score have been used as classification quality measures. These corpora are divided into the training (60%) and the test (40%) sets by the organizers of the campaign and this partition has been kept in this study.

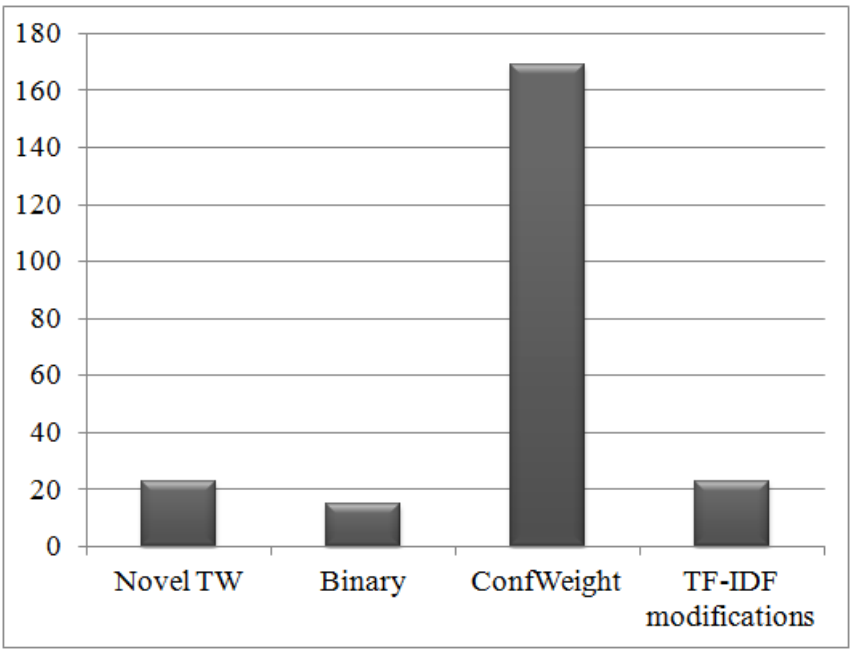

Figure 5.3: Computational time (min) obtained using different text preprocessing methods on the SpeechCycle corpus (the TF-IDF modifications time is represented by the average values of all TF-IDF modifications)

RapidMiner has been used to perform text classification. Implemented classification methods are:

- k-nearest neighbors method with weighted vote (k varies from 1 to 15);
- kernel Bayes classifier with Laplace correction;
- linear Support Vector Machine;
- Artificial Neural Network (ANN) with error back propagation learning.

First, for each corpus and for each term weighting algorithm a set of 10 filter levels (borderline values) has been calculated by extensive numerical experiments. These numerical experiments have been conducted using the Baden-Württemberg Grid (bwGRiD) Cluster Ulm (Every blade comprehends two 4-Core Intel Harpertown CPUs with 2.83 GHz and 16 GByte RAM). The selected borderline values for different preprocessing techniques are presented in Table 5.8.

Tables 5.9 and 5.13 present the F-scores obtained on the test corpora. The best values are shown in bold. Results of the k-NN algorithm are presented with the best k parameter.

Experimental results obtained on the reduced term set in comparison with the initial feature space on the Books corpus are shown in Table 5.9. The best performance has been achieved using the initial term set, artificial neural network as a text classifier and ConfWeight

Corpus	**Pre processing**	**Borderline values**									
Books	Novel TW	0.01	0.03	0.08	0.1	0.13	0.15	0.18	0.2	0.23	0.25
	Conf Weight	0.02	0.03	0.05	0.07	0.13	0.17	0.18	0.19	0.23	0.25
	tf.idf 1	0.02	0.04	0.05	0.06	0.07	0.08	0.14	0.16	0.17	0.19
	tf.idf 2	0.01	0.03	0.04	0.06	0.08	0.09	0.14	0.16	0.17	0.25
Games	Novel TW	0.01	0.03	0.08	0.1	0.13	0.15	0.2	0.25	0.3	0.35
	Conf Weight	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
	tf.idf 1	0.01	0.05	0.08	0.13	0.18	0.2	0.25	0.28	0.3	0.35
	tf.idf 2	0.01	0.03	0.04	0.07	0.08	0.11	0.12	0.15	0.16	0.19
Debates	Novel TW	0.02	0.05	0.08	0.1	0.15	0.2	0.23	0.25	0.3	0.35
	Conf Weight	0.01	0.05	0.09	0.1	0.13	0.15	0.17	0.2	0.25	0.3
	tf.idf 1	0.01	0.05	0.09	0.13	0.17	0.2	0.25	0.28	0.3	0.35
	tf.idf 2	0.02	0.05	0.08	0.1	0.15	0.2	0.25	0.28	0.3	0.35
T1	Novel TW	0.02	0.05	0.08	0.1	0.15	0.2	0.25	0.28	0.3	0.35
	Conf Weight	0.01	0.05	0.09	0.13	0.17	0.2	0.25	0.28	0.3	0.35
	tf.idf 1	0.01	0.05	0.08	0.1	0.15	0.18	0.2	0.25	0.3	0.35
	tf.idf 2	0.01	0.05	0.09	0.13	0.17	0.2	0.25	0.28	0.3	0.35
T2	Novel TW	0.02	0.05	0.08	0.1	0.15	0.2	0.25	0.28	0.3	0.35
	Conf Weight	0.01	0.05	0.08	0.12	0.17	0.2	0.25	0.28	0.3	0.35
	tf.idf 1	0.01	0.05	0.08	0.12	0.17	0.2	0.23	0.26	0.29	0.35
	tf.idf 2	0.01	0.05	0.1	0.15	0.2	0.23	0.26	0.29	0.35	

Table 5.8: List of borderline values applied for dimensionality reduction

	Reduced term set				**Initial term set**			
	ANN (filter)	SVM (filter)	k-NN (filter) (k)	Bayes (filter)	ANN	SVM	k-NN (k)	Bayes
Novel TW	0.485 (0.03)	0.490 (0.03)	0.493 (0.08) (15)	0.238 (0.03)	0.493	**0.516**	0.488 (14)	0.437
Conf Weight	**0.546** (0.03)	0.238 (0.03)	**0.544** (0.05) (12)	0.238 (0.03)	**0.570**	0.238	**0.559** (15)	0.238
tf.idf 1	0.499 (0.04)	**0.527** (0.04)	0.504 (0.16) (15)	**0.516** (0.04)	0.505	0.499	0.517 (3)	0.495
tf.idf 2	0.509 (0.06)	0.525 (0.17)	0.504 (0.25) (15)	0.508 (0.08)	0.505	0.511	0.516 (1)	**0.506**

Table 5.9: Experimental results obtained on the Books corpus using the filtered term set and the initial term set

as a text preprocessing technique ($Fscore = 0.570$). Using the reduced feature space the highest F-score has been also obtained by artificial neural network and ConfWeight ($Fscore = 0.5455$). The best combinations of text preprocessing techniques and classification methods are ConfWeight with ANN and ConfWeight with k-NN.

Table 5.10 presents the numerical experiments conducted on the Games corpus using the initial and the reduced feature space. On this corpus the ANN classifier and the ConfWeight preprocessing using the reduced term set have provided the highest F-score ($Fscore = 0.7307$). The performance on the initial term set has reached 0.7203 with the ConfWeight and k-nearest neighbors algorithm ($k = 15$). Table 5.10 shows that the ConfWeight combined with the ANN classifier or k-NN algorithm produces significantly higher performance using both the initial and the reduced feature spaces.

The comparison of performances obtained on the initial and the reduced term sets of the Debates corpus is presented in Table 5.11. The highest F-score (0.7101) has been obtained using the reduced feature space, ConfWeight and the ANN classifier. Using the initial feature space, ConfWeight and the ANN classifier the best performance has achieved 0.7051. The best combinations obtained on the Debates corpus are ConfWeight with ANN, ConfWeight with k-NN and novel TW with the SVM classifier.

Table 5.12 compares the experimental results conducted on the T1 corpus using the initial and the reduced feature space. The best performance has reached 0.8571 on the reduced term set, ConfWeight as a text preprocessing method and ANN as a classification algorithm. The ANN classifier and novel TW provide the highest F-score ($Fscore = 0.8535$) on the initial

	Reduced term set				**Initial term set**			
	ANN (filter)	SVM (filter)	k-NN (filter) (k)	Bayes (filter)	ANN	SVM	k-NN (k)	Bayes
Novel TW	0.675 (0.01)	**0.684** (0.01)	0.695 (0.03) (11)	0.210 (0.03)	0.691	**0.675**	0.699 (13)	**0.675**
Conf Weight	**0.731** (0.02)	0.210 (0.03)	**0.731** (0.01) (14)	0.210 (0.03)	**0.717**	0.210	**0.720** (15)	0.210
tf.idf 1	0.687 (0.1)	0.669 (0.25)	0.693 (0.01) (4)	0.681 (0.13)	0.677	0.665	0.672 (3)	0.652
tf.idf 2	0.679 (0.04)	0.677 (0.08)	0.696 (0.16) (5)	**0.684** (0.19)	0.664	0.661	0.671 (7)	0.651

Table 5.10: Experimental results obtained on the Games corpus using the filtered term set and the initial term set

	Reduced term set				**Initial term set**			
	ANN (filter)	SVM (filter)	k-NN (filter) (k)	Bayes (filter)	ANN	SVM	k-NN (k)	Bayes
Novel TW	0.698 (0.1)	**0.699** (0.1)	0.694 (0.02) (15)	0.363 (0.01)	0.697	**0.702**	0.695 (15)	0.616
Conf Weight	**0.710** (0.01)	0.637 (0.01)	**0.699** (0.01) (13)	0.363 (0.01)	**0.705**	0.634	**0.695** (15)	0.363
tf.idf 1	0.680 (0.05)	0.678 (0.2)	0.675 (0.13) (15)	**0.673** (0.01)	0.638	0.642	0.637 (15)	0.637
tf.idf 2	0.678 (0.15)	0.676 (0.1)	0.671 (0.3) (11)	0.670 (0.28)	0.632	0.640	0.634 (15)	**0.639**

Table 5.11: Experimental results obtained on the Debates corpus using the filtered term set and the initial term set

	Reduced term set				**Initial term set**			
	ANN (filter)	SVM (filter)	k-NN (filter) (k)	Bayes (filter)	ANN	SVM	k-NN (k)	Bayes
Novel TW	0.846 (0.02)	**0.838** (0.25)	0.829 (0.2) (12)	0.753 (0.05)	**0.854**	0.834	0.837 (13)	0.794
Conf Weight	**0.857** (0.01)	0.835 (0.05)	**0.850** (0.09) (13)	0.529 (0.01)	0.853	**0.848**	**0.855** (14)	**0.837**
tf.idf 1	0.818 (0.3)	0.810 (0.18)	0.811 (0.35) (11)	0.803 (0.01)	0.830	0.804	0.816 (15)	0.591
tf.idf 2	0.817 (0.3)	0.810 (0.3)	0.810 (0.2) (15)	**0.807** (0.35)	0.808	0.812	0.808 (15)	0.690

Table 5.12: Experimental results obtained on the T1 corpus using the filtered term set and the initial term set

term set. Table 5.12 shows that ConfWeight combined with ANN or k-NN, novel TW with ANN perform well on the reduced feature space, ConfWeight combined with ANN, SVM and k-NN, as well as novel TW with ANN produce good performance on the initial feature space.

Numerical experiments conducted on the T2 corpus using the reduced term set in comparison with the initial feature space are presented in Table 5.13. Using the reduced term set, ANN and novel TW the best performance has reached 0.8575. The SVM classification algorithm combined with the novel TW provides the highest F-score ($Fscore = 0.8520$) on the initial term set. On the T2 corpus novel TW produces the highest F-scores in comparison with all other text preprocessing techniques. The best test classifiers are the ANN and the SVM algorithms.

Table 5.14 provides an overall comparison of classification performance using the initial and the reduced feature space. It shows the best F-scores obtained on the test data of the Books, Games, Debates, T1 and T2 corpora.

Table 5.14 shows that using the filter level of 0.03 the best performance obtained on the Books corpus ($Fscore = 0.546$) is achieved with the ConfWeight as a text preprocessing technique and the artificial neural network as a classifier, however, it is lower than the result on the initial feature space. On the Games corpus the performance obtained using the reduced term set ($borderline = 0.01$) is greater than the one using the initial term set (initial: $Fscore = 0.720$, reduced: $Fscore = 0.731$). Using the filter level of 0.01 on the Debates and T1 corpora the results obtained with the reduced feature space (Debates: $Fscore = 0.710$, T1: $Fscore = 0.857$) outperform the results with the initial feature space

	Reduced term set				**Initial term set**			
	ANN (filter)	SVM (filter)	k-NN (filter) (k)	Bayes (filter)	ANN	SVM	k-NN (k)	Bayes
Novel TW	**0.858** (0.1)	**0.853** (0.1)	**0.811** (0.05) (14)	**0.749** (0.05)	**0.843**	**0.852**	**0.811** (12)	**0.745**
Conf Weight	0.822 (0.01)	0.815 (0.01)	0.784 (0.05) (14)	0.710 (0.01)	0.820	0.815	0.785 (15)	0.712
tf.idf 1	0.838 (0.17)	0.830 (0.35)	0.785 (0.17) (13)	0.660 (0.05)	0.838	0.830	0.786 (15)	0.658
tf.idf 2	0.826 (0.26)	0.830 (0.23)	0.775 (0.2) (15)	0.728 (0.26)	0.826	0.830	0.775 (14)	0.728

Table 5.13: Experimental results obtained on the T2 corpus using the filtered term set and the initial term set

Corpus	**Initial term set**			**Reduced term set**			
	F-score	Prepro cessing	Classifier	F-score	Prepro cessing	Classifier	Borderline
Books	**0.570**	Conf Weight	ANN	0.546	Conf Weight	ANN	0.03
Games	0.720	Conf Weight	k-NN ($k = 15$)	**0.731**	Conf Weight	ANN	0.01
Debates	0.705	Conf Weight	ANN	**0.710**	Conf Weight	ANN	0.01
T1	0.855	Conf Weight	k-NN ($k = 14$)	**0.857**	Conf Weight	ANN	0.01
T2	0.852	Novel TW	SVM	**0.858**	Novel TW	ANN	0.1

Table 5.14: Overall comparison of the performance obtained using the initial and the reduced term sets

(Debates: $Fscore = 0.705$, T1: $Fscore = 0.855$). On the T2 corpus the best performance has been obtained using the borderline 0.1, novel TW as a text preprocessing technique and artificial neural network as a text classifier ($Fscore = 0.858$), which outperforms the result achieved using the initial term set ($Fscore = 0.852$). After feature selection the best performance has been obtained using artificial neural networks. The experimental results have shown that ConfWeight provides the highest Fscore on the Books, Games, Debates and T1 corpora and novel TW outperforms other text preprocessing techniques on the T2 corpus.

The experimental results using a feature selection algorithm applied in this thesis for text classification have been presented in this section. This approach based on the term filtering has been tested using 10 filter levels selected for each corpus and text preprocessing separately, different term weighting techniques (two modifications of TF-IDF, Confidence Weights, and the Novel TW), standard classification algorithms (Bayes classifier, k-NN, SVM, and artificial neural network), two text classification tasks (opinion mining: Books, Games and Debates corpora; topic categorization: T1 and T2 corpora). It can be concluded that the reduced feature space obtained by this approach produces better or similar performance with the lower dimensionality and less CPU's processing time.

5.3.2 Reduced Feature Space using Feature Extraction

The proposed method reduces the term set combining term clustering (the hierarchical agglomerative clustering algorithm) with heuristic optimization of the weights of the obtained term clusters (the cooperative scheme of the coevolutionary genetic algorithm). The settings of the coevolutionary genetic algorithm which have been used in the simulations conducted in this thesis are presented in Table 5.15.

Since all applied classification algorithms tend to quickly overfit on the Books and Games corpora, the fitness function of the genetic algorithm is modified to take advantage from the unlabelled data. This semi-supervised modification of the fitness function is given by

$$\text{fitness} = \begin{cases} \text{external}_{train}, & \text{if external}_{train} < 0.95) \\ \text{external}_{train} + \text{internal}_{train+test}, & \text{otherwise} \end{cases} \rightarrow \max, \tag{5.1}$$

where N is the number of clusters, c_i is a centroid of the cluster i, c_j is a centroid of the cluster j and external_{train} is an accuracy obtained on the training set according to the decision rule $\text{class winner} = \arg\left(\max_i \sum_{j:S_j=i} C_j\right)$.

The external criterion is given by

Parameter	Value
Subpopulation size	100
Number of generations when subpopulations work separately	5
Number of iterations	15
Number of bits to code each variable	17
Variables lie in the interval	[0;1]
Step	0,00000763
Selection	Tournament with the size = 3
Crossover	Uniform
Mutation	$\text{Probability} = \frac{1}{\text{Number of bits} * \text{Number of clusters}}$

Table 5.15: Settings of the coevolutionary genetic algorithm

$$\text{external}_{train} = \sum_{k=1}^{K} \delta_k \rightarrow \max, \tag{5.2}$$

where δ_k is defined as

$$\delta_k = \begin{cases} 1, & \text{if } \arg\left(\max_i \sum_{j:S_j=i} w_j\right) = \text{prediction}_k \\ 0, & \text{otherwise} \end{cases}. \tag{5.3}$$

The internal criterion is given by

$$\text{internal}_{train+test} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \|c_i - c_j\| \rightarrow \max, \tag{5.4}$$

where c_i is a centroid of the cluster i obtained with this classifier on both labelled and unlabelled data, N is the number of categories, w_j is a weight of the term j, S_j is the number of category where the term j has the highest relative frequency of occurrence and prediction_k is the label assigned to the k object by the classifier.

5.3.2.1 Opinion Mining Corpora

In this section the experimental results obtained on the DEFT 2007 corpora (Books, Games and Debates) are described.

Implemented classification methods are:

- k-nearest neighbors method with weighted vote (k varies from 1 to 15);
- kernel Bayes classifier with Laplace correction;
- Support Vector Machine with ML;
- Support Vector Machine optimized using COBRA;
- Rocchio classifier;
- decision rule 4.3;
- Artificial Neural Network (ANN) with error back propagation learning;
- Artificial Neural Network (ANN) optimized using COBRA.

Tables 5.16 and 5.21 present the F-scores obtained on the test corpora using the semi-supervised dimensionality reduction method based on the term clustering. F-scores obtained on the test corpora are considered as classification quality measures. The best values are shown in bold. Results of the k-NN algorithm are presented with the best k parameter.

Detailed numerical results obtained on the Books corpus using the semi-supervised dimensionality reduction method with 100 term clusters for each category with standard classification algorithms are presented in Table 5.16. The best classification quality ($Fscore = 0.613$) is provided with the tf.idf 4 $\gamma = 0.1$ as a text preprocessing technique and the ANN generated using COBRA as a classification method. This result outperforms the best one obtained by DEFT'07 participants (best $Fscore = 0.603$, average $Fscore = 0.500$) and the performance obtained on the feature space reduced using the feature selection approach ($Fscore = 0.546$). The SVM algorithm generated using COBRA on the initial feature space ($Fscore = 0.619$) outperforms the result on the reduced term set, however, this semi-supervised text preprocessing requires less CPU's computational time in order to obtain this relatively good performance.

The performance of standard classification algorithms on the Books corpus combined with different semi-supervised text preprocessing techniques using 50 term clusters for each category is shown in Table 5.17. The best classification quality ($Fscore = 0.612$) is provided with the tf.idf 4 ($\gamma = 0.5$) as a text preprocessing technique and the ANN generated using COBRA as a classification method. This result outperforms the best one obtained by DEFT'07 participants (best $Fscore = 0.603$, average $Fscore = 0.500$) and the performance obtained on the feature space reduced using the feature selection approach ($Fscore = 0.546$). The SVM algorithm generated using COBRA on the initial feature space ($Fscore = 0.619$) and on the reduced feature space with 100 term clusters for each category ($Fscore = 0.613$) outperforms the result on the reduced term set with 50 term clusters for each category, however, this semi-supervised text preprocessing requires less CPU's computational time in order to obtain this relatively good performance.

	k-**NN**	**Bayes**	**SVM** RM	**SVM** CO-BRA	**Roc chio**	**Dec. rule 4.3**	**ANN** RM	**ANN** CO-BRA
Novel TW	0,526 (11)	0,499	0,519	**0,594**	0,519	0,539	0,512	0,587
Conf Weight	0,455 (11)	0,432	0,444	0,588	0,455	**0,566**	0,455	0,455
tf.idf 1	0,507 (11)	0,492	0,517	0,586	0,528	0,530	0,504	0,556
tf.idf 2	0,521 (13)	0,500	**0,536**	0,561	0,525	0,537	**0,534**	0,607
tf.idf 3 $\gamma = 0.1$	**0,536 (15)**	0,509	0,526	0,587	**0,545**	0,534	0,517	0,551
tf.idf 3 $\gamma = 0.5$	0,527 (4)	**0,512**	0,528	0,566	0,539	0,521	0,511	0,571
tf.idf 3 $\gamma = 0.9$	0,528 (2)	0,512	0,522	0,576	0,532	0,517	0,520	0,570
tf.idf 4 $\gamma = 0.1$	0,521 (15)	0,491	0,510	0,556	0,514	0,516	0,506	**0,613**
tf.idf 4 $\gamma = 0.5$	0,524 (6)	0,515	0,529	0,537	0,525	0,527	0,494	0,574
tf.idf 4 $\gamma = 0.9$	0,523 (6)	0,494	0,513	0,577	0,513	0,519	0,519	0,543

Table 5.16: F-scores obtained using the semi-supervised dimensionality reduction method (100 term clusters per category): Books

Figure 5.4 compares the best performances obtained using different text preprocessing algorithms with the initial and reduced (100 and 50 term clusters per category) term sets on the Books corpus. It has been shown that the performance of the classification algorithms obtained on the reduced term sets becomes worse with the Novel TW, ConfWeight and tf.idf 3 $\gamma = 0.5$ text representations, however, text preprocessing obtained on the reduced term set has provided better algorithmic performance with tf.idf 1, tf.idf 2, tf.idf 3 $\gamma = 0.1; 0.9$ and tf.idf 4 $\gamma = 0.1; 0.5; 0.9$. Using 50 term clusters per category in 7 cases (ConfWeight, tf.idf 1, if.idf 2, tf.idf 3 $\gamma = 0.1; 0.5; 0.9$, tf.idf 4$\gamma = 0.9$) produces worse representations than with 100 term clusters per category and in 3 cases (novel TW, tf.idf 4$\gamma = 0.1; 0.5$) it provided better representation. On this corpus the best results have been obtained using the novel TW preprocessing on the initial feature space; the best performance on the reduced feature space has been achieved with the novel TW, tf.idf 2 and tf.idf 4 $\gamma = 0.5$ representations. These results show that the supervised text preprocessing techniques such as the novel TW and ConfWeight produce better performance with the initial feature space, however, even so unsupervised tf.idf modifications do not generally perform well on this corpus the tf.idf 2

	k-NN	Bayes	SVM RM	SVM CO-BRA	Rocchio	Dec. rule 4.3	ANN RM	ANN CO-BRA
Novel TW	**0,538 (9)**	**0,504**	0,505	**0,602**	**0,536**	0,517	**0,515**	0,584
Conf Weight	0,445 (14)	0,450	0,429	0,556	0,479	**0,560**	0,435	0,429
tf.idf 1	0,500 (14)	0,487	0,500	0,556	0,513	0,517	0,498	0,549
tf.idf 2	0,513 (3)	0,484	0,495	0,570	0,495	0,506	0,483	0,567
tf.idf 3 $\gamma = 0.1$	0,500 (1)	0,489	**0,521**	0,460	0,516	0,517	0,503	0,564
tf.idf 3 $\gamma = 0.5$	0,486 (14)	0,458	0,480	0,566	0,492	0,488	0,462	0,557
tf.idf 3 $\gamma = 0.9$	0,508 (9)	0,503	0,504	0,547	0,511	0,519	0,489	0,571
tf.idf 4 $\gamma = 0.1$	0,512 (14)	0,468	0,492	0,577	0,516	0,518	0,487	0,546
tf.idf 4 $\gamma = 0.5$	0,513 (15)	0,484	0,471	0,558	0,514	0,517	0,481	**0,612**
tf.idf 4 $\gamma = 0.9$	0,501 (15)	0,482	0,489	0,568	0,508	0,495	0,483	0,561

Table 5.17: F-scores obtained using the semi-supervised dimensionality reduction method (50 term clusters per category): Books

using 100 term clusters and tf.idf 4 $\gamma = 0.5$ using only 50 term clusters both in combination with the ANN text classifier generated using COBRA achieve results close to the best ones obtained using the supervised novel TW and ConfWeight.

Table 5.18 presents the detailed results of the numerical experiments conducted on the Games corpus using the feature space dimensionality reduction method with 100 term clusters for each category in combination with standard text classification algorithms. The k-NN algorithm ($k = 13$) combined with the tf.idf 4 ($\gamma = 0.1$) preprocessing technique outperforms the results obtained by other combinations ($Fscore = 0.731$). This result is better than the average performance obtained by DEFT'07 participants (average $Fscore = 0.660$, best $Fscore = 0.784$) and outperforms the result on the initial feature space ($Fscore = 0.720$) even so it requires more computational time. The obtained performance also matches the result produced by the term filtering approach. The numerical experiments have shown that on the Games corpus the best text preprocessing techniques are the novel TW and the tf.idf 4 with $\gamma = 0.1$.

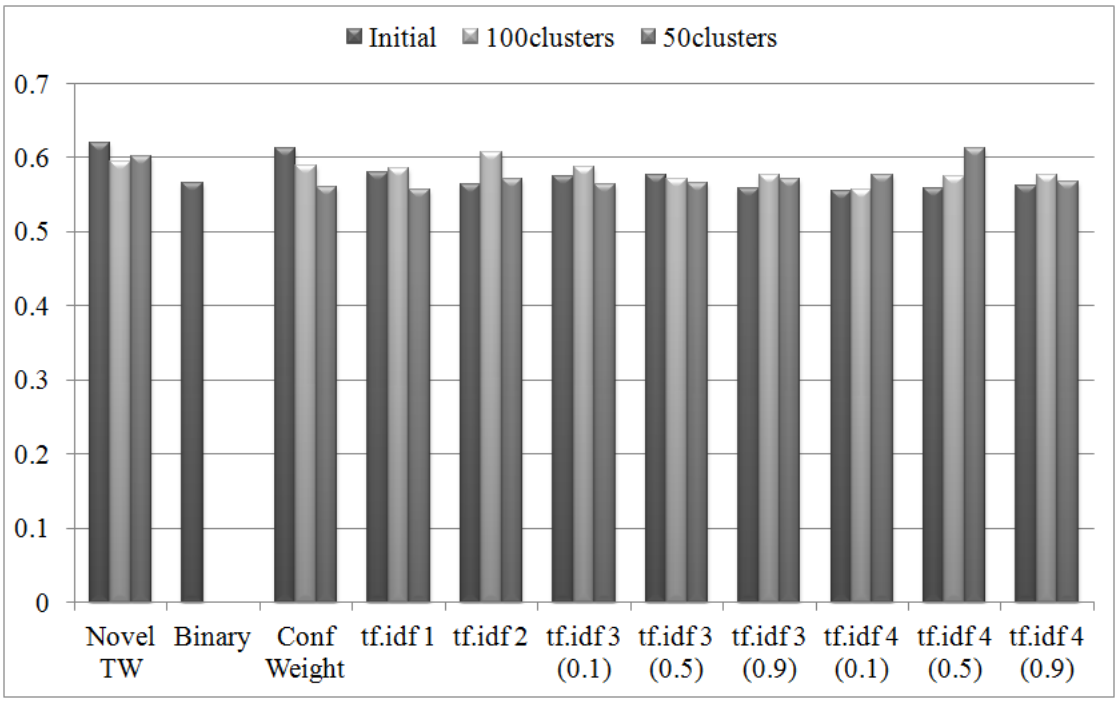

Figure 5.4: The best algorithmic performance obtained on the Books corpus using different text preprocessings

The numerical experiments conducted on the Games corpus using several text classifiers combined with different semi-supervised text preprocessing techniques using 50 term clusters for each category are shown in Table 5.19. The best classification quality ($Fscore = 0.722$) is provided by the combination of the novel TW preprocessing and Rocchio classifier and by the combination of the tf.idf 4 ($\gamma = 0.1$) preprocessing and the k-NN classification algorithm. This result outperforms the average F-score (average $Fscore = 0.660$,the best $Fscore = 0.720$) obtained by the participants of the DEFT 2008. The semi-supervised dimensionality reduction algorithm with 100 term clusters per category ($Fscore = 0.731$) and the F-score obtained on the feature space reduced using the feature selection approach ($Fscore = 0.731$) produce however better classification quality. On this corpus the novel TW preprocessing outperforms other term weighting methods in combination with all implemented classifiers excluding the k-NN algorithm that reaches the highest F-score with the tf.idf 4 ($\gamma = 0.1$).

The best performances obtained using different text preprocessing algorithms on the corpus Games are presented in Figure 5.5. It has been shown that the classification quality on the reduced term sets has been is better than the results on the initial term set with all text representations excluding the ConfWeight. The use of 50 term clusters per category in 9 cases (ConfWeight, tf.idf 1, if.idf 2, tf.idf 3 $\gamma = 0.1; 0.5; 0.9$, tf.idf 4$\gamma = 0.1; 0.5; 0.9$) produces worse text representations than the use of 100 term clusters per category and in the case of the

	k-**NN**	**Bayes**	**SVM** RM	**SVM** CO-BRA	**Rocchio**	**Dec. rule 4.3**	**ANN** RM	**ANN** CO-BRA
Novel TW	0,723 (10)	**0,682**	**0,710**	**0,668**	0,716	0,717	0,708	0,706
Conf Weight	0,587 (15)	0,557	0,557	0,645	0,590	0,695	0,570	0,717
tf.idf 1	0,710 (11)	0,662	0,685	0,635	0,708	0,698	0,706	0,702
tf.idf 2	0,723 (12)	0,657	0,698	0,613	0,718	0,709	0,705	**0,725**
tf.idf 3 $\gamma = 0.1$	0,717 (14)	0,670	0,699	0,640	0,716	0,710	0,700	0,674
tf.idf 3 $\gamma = 0.5$	0,705 (15)	0,660	0,697	0,637	0,708	0,702	0,710	0,672
tf.idf 3 $\gamma = 0.9$	0,698 (5)	0,653	0,683	0,607	0,691	0,686	0,695	0,685
tf.idf 4 $\gamma = 0.1$	**0,731 (13)**	0,661	0,681	0,634	**0,726**	**0,722**	**0,721**	0,715
tf.idf 3 $\gamma = 0.5$	0,718 (13)	0,666	0,696	0,632	0,706	0,708	0,712	0,700
tf.idf 4 $\gamma = 0.9$	0,707 (9)	0,656	0,689	0,655	0,703	0,700	0,708	0,696

Table 5.18: F-scores obtained using the semi-supervised dimensionality reduction method (100 term clusters per category): Games

novel TW this further reduction of the feature space has not influenced the performance. On this corpus similar results have been achieved using the novel TW, ConfWeight, tf.idf 2, tf.idf 4 $\gamma = 0.1$. The best performance has been obtained using the reduced term set with 100 clusters per category, tf.idf 4 $\gamma = 0.1$ and the k-NN algorithm.

Using the feature space dimensionality reduction approach based on the semi-supervised term clustering (100 term clusters per category) on the Debates corpus the best classification quality ($Fscore = 0.704$) has been obtained with the ConfWeight as a term weighting algorithm and the SVM generated using COBRA as a classification method. This result outperforms the average result of the DEFT 2008 participants (average $Fscore = 0.642$, the best $Fscore = 0.720$). The F-scores obtained using the initial feature space ($Fscore = 0.714$) and using the reduced feature space after applying the term filtering algorithm ($Fscore = 0.710$) are nevertheless higher. Table 5.20 shows that the majority of implemented text classifiers work better in combination with the ConfWeight term weighting. The k-NN algorithm and the Bayes classifier produce the highest F-scores in combination with the novel TW and the tf.idf 4 ($\gamma = 0.9$) respectively.

	k-**NN**	**Bayes**	**SVM** RM	**SVM** CO-BRA	**Rocchio**	**Dec. rule 4.3**	**ANN** RM	**ANN** CO-BRA
Novel TW	0,717 (13)	**0,691**	**0,708**	**0,681**	**0,722**	**0,716**	**0,713**	**0,712**
Conf Weight	0,587 (12)	0,541	0,563	0,625	0,569	0,699	0,588	0,705
tf.idf 1	0,695 (15)	0,653	0,671	0,638	0,686	0,681	0,696	0,693
tf.idf 2	0,712 (5)	0,652	0,687	0,630	0,707	0,704	0,709	0,694
tf.idf 3 $\gamma = 0.1$	0,689 (15)	0,651	0,681	0,639	0,685	0,686	0,685	0,667
tf.idf 3 $\gamma = 0.5$	0,682 (12)	0,635	0,676	0,632	0,686	0,676	0,681	0,674
tf.idf 3 $\gamma = 0.9$	0,642 (4)	0,618	0,625	0,622	0,641	0,642	0,640	0,676
tf.idf 4 $\gamma = 0.1$	**0,722 (14)**	0,652	0,702	0,653	0,708	0,706	0,712	0,694
tf.idf 3 $\gamma = 0.5$	0,701 (9)	0,666	0,689	0,651	0,689	0,693	0,691	0,702
tf.idf 4 $\gamma = 0.9$	0,665 (8)	0,640	0,650	0,634	0,664	0,654	0,657	0,662

Table 5.19: F-scores obtained using the semi-supervised dimensionality reduction method (50 term clusters per category): Games

The classification performance obtained on the Debates corpus using the semi-supervised text preprocessing techniques with the reduced term set (50 term clusters for each category) combined with standard text classifiers is shown in Table 5.21. The novel TW as a term weighting algorithm and the Rocchio classifier produce the best classification quality ($Fscore = 0.712$) on the Debates corpus. This result outperforms the average F-score (average $Fscore = 0.642$,the best $Fscore = 0.720$) obtained by the participants of the DEFT 2008 and it also outperforms the result produced by the semi-supervised dimensionality reduction algorithm with 100 term clusters per category ($Fscore = 0.704$) as well as the F-score obtained on the feature space reduced using the feature selection approach ($Fscore = 0.710$). On this corpus the ConfWeight preprocessing outperforms other term weighting methods in combination with all implemented classifiers excluding the Bayes algorithm that reaches the highest F-score with the novel TW.

The results of the numerical experiments on the corpus Debates conducted in this thesis are presented in Figure 5.6. The performance of the classification algorithms obtained on the

	k-NN	**Bayes**	**SVM** RM	**SVM** CO-BRA	**Rocchio**	**Dec. rule 4.3**	**ANN** RM	**ANN** CO-BRA
Novel TW	**0,692 (15)**	0,593	0,695	0,659	0,693	0,695	0,695	0,700
Conf Weight	0,688 (15)	0,558	**0,697**	**0,704**	**0,698**	**0,701**	**0,699**	**0,702**
tf.idf 1	0,679 (14)	0,599	0,677	0,628	0,675	0,677	0,682	0,704
tf.idf 2	0,679 (15)	0,592	0,683	0,649	0,682	0,682	0,684	0,690
tf.idf 3 $\gamma = 0.1$	0,678 (14)	0,568	0,682	0,646	0,668	0,676	0,681	0,684
tf.idf 3 $\gamma = 0.5$	0,672 (11)	0,599	0,669	0,663	0,662	0,670	0,674	0,675
tf.idf 3 $\gamma = 0.9$	0,663 (12)	0,593	0,667	0,640	0,652	0,662	0,663	0,673
tf.idf 4 $\gamma = 0.1$	0,664 (11)	0,549	0,666	0,609	0,667	0,674	0,668	0,674
tf.idf 3 $\gamma = 0.5$	0,670 (13)	0,579	0,670	0,645	0,661	0,668	0,669	0,654
tf.idf 4 $\gamma = 0.9$	0,659 (15)	**0,600**	0,661	0,643	0,658	0,660	0,661	0,663

Table 5.20: F-scores obtained using the semi-supervised dimensionality reduction method (100 term clusters per category): Debates

	k-**NN**	**Bayes**	**SVM** RM	**SVM** CO-BRA	**Roc chio**	**Dec. rule 4.3**	**ANN** RM	**ANN** CO-BRA
Novel TW	0,690 (14)	**0,611**	0,694	0,668	0,688	0,690	0,691	0,698
Conf Weight	**0,703 (15)**	0,607	**0,705**	**0,708**	**0,712**	**0,704**	**0,709**	**0,706**
tf.idf 1	0,678 (13)	0,606	0,679	0,661	0,668	0,672	0,675	0,700
tf.idf 2	0,666 (11)	0,583	0,671	0,620	0,662	0,672	0,670	0,664
tf.idf 3 $\gamma = 0.1$	0,673 (15)	0,577	0,679	0,653	0,664	0,675	0,676	0,683
tf.idf 3 $\gamma = 0.5$	0,667 (11)	0,593	0,667	0,633	0,662	0,562	0,665	0,690
tf.idf 3 $\gamma = 0.9$	0,647 (9)	0,583	0,660	0,630	0,645	0,647	0,647	0,654
tf.idf 4 $\gamma = 0.1$	0,671 (13)	0,581	0,677	0,641	0,670	0,676	0,676	0,646
tf.idf 3 $\gamma = 0.5$	0,661 (14)	0,602	0,661	0,652	0,654	0,659	0,662	0,668
tf.idf 4 $\gamma = 0.9$	0,653 (15)	0,575	0,654	0,642	0,647	0,655	0,655	0,663

Table 5.21: F-scores obtained using the semi-supervised dimensionality reduction method (50 term clusters per category): Debates

Figure 5.5: The best algorithmic performance obtained on the Games corpus using different text preprocessings

reduced term sets improves with the use of tf.idf 1, tf.idf 2, tf.idf 3 and tf.idf 4 $\gamma = 0.1$. In cases of the novel TW and ConfWeight the classification quality becomes worse and using the tf.idf 4 $\gamma = 0.5; 0.9$ the obtained performance does not change significantly. Using 50 term clusters per category improves the performance in 3 cases (ConfWeight, tf.idf 3 $\gamma = 0.5$, tf.idf 4 $\gamma = 0.1$); in 3 cases (tf.idf 1, if.idf 2, tf.idf 3 $\gamma = 0.9$) produces worse representations than with 100 term clusters per category and in 4 cases (novel TW, tf.idf 3 $\gamma = 0.1$ and tf.idf 4 $\gamma = 0.5; 0.9$) the use of less term clusters did not have an impact on the performance. On this corpus the best performance has been achieved using the ConfWeight representation combined with the SVM text classifier generated using COBRA; however the results obtained on the novel TW in combination with the ANN optimized using COBRA are close enough to the ConfWeight.

The comparison of the best combinations of the proposed methods obtained using the initial and reduced (100 and 50 term clusters per category) term sets with the results of the participants of DEFT 2007 is shown in Table 5.22. The best performance has been obtained by Torres-Moreno team for the Games and Debates corpora ($Fscore = 0.784$ and $Fscore = 0.720$) and by the combination introduced in this thesis (the novel TW as a text preprocessing technique and the SVM generated using COBRA as a classification method) for the Books corpus ($Fscore = 0.619$). The best combinations proposed in this thesis have

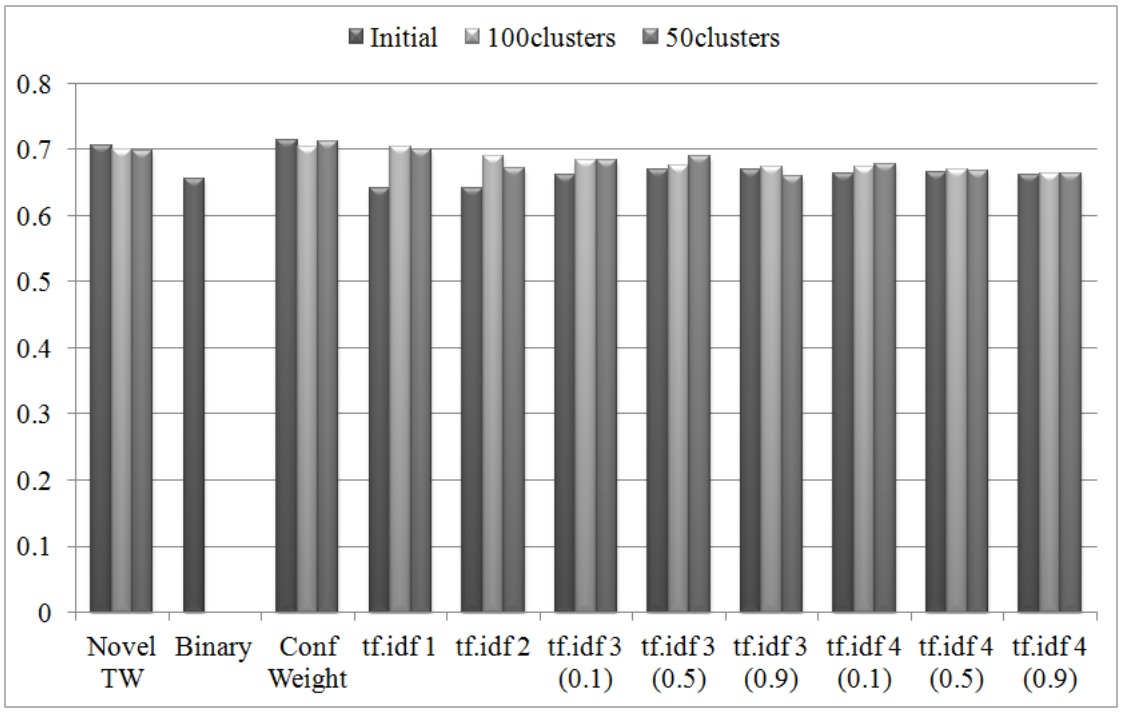

Figure 5.6: The best algorithmic performance obtained on the Debates corpus using different text preprocessings

reached 0.731 and 0.714 for the Games and Debates corpora using the reduced feature space obtained with the term filtering approach for the Games corpus (4th place) and the initial feature space for the Debates corpus (2nd place).

5.3.2.2 Document Classification

In this section the experimental results obtained on the DEFT 2008 corpora (T1 and T2) are described. Both databases are designed for topic categorization tasks, have significantly greater number of documents and greater dimensionality of the feature space.

Implemented classification methods are:

- k-nearest neighbors method with weighted vote (k varies from 1 to 15);
- kernel Bayes classifier with Laplace correction;
- linear Support Vector Machine;
- Support Vector Machine with ML;
- Support Vector Machine optimized using COBRA;
- Rocchio classifier;

Team	Books	Games	Debates	Rank
J.-M. Torres-Moreno (LIA)	0,603 (4)	0,784 (1)	0,720 (1)	1
G. Denhiere (EPHE et U. Wurzburg)	0,599 (5)	0,699 (9)	0,681 (8)	7
S. Maurel (CELI France)	0,519 (10)	0,706 (8)	0,697 (7)	9
M. Vernier (GREYC)	0,577 (6)	0,761 (3)	0,673 (10)	6
E. Crestan (Yahoo ! Inc.)	0,529 (9)	0,673 (10)	0,703 (5)	8
M. Plantie (LGI2P et LIRMM)	0,472 (12)	0,783 (2)	0,671 (11)	9
A.-P. Trinh (LIP6)	0,542 (8)	0,659 (11)	0,676 (9)	10
M. Genereux (NLTG)	0,464 (13)	0,626 (12)	0,569 (14)	12
E. Charton (LIA)	0,504 (11)	0,619 (13)	0,616 (12)	11
A. Acosta (Lattice)	0,392 (14)	0,536 (14)	0,582 (13)	13
Initial term set	**0,619 (1)**	**0,720 (7)**	**0,714 (2)**	**2**
Reduced term set using feature selection	**0,546 (7)**	**0,731 (4)**	**0,710 (4)**	**5**
Reduced term set using feature extraction (100 term clusters for each category)	**0,613 (2)**	**0,731 (5)**	**0,704 (6)**	**4**
Reduced term set using feature extraction (50 term clusters for each category)	**0,612 (3)**	**0,722 (6)**	**0,712 (3)**	**3**
Average result obtained by participants	0,500	0,660	0,642	

Table 5.22: Total results on the corpora DEFT 2007 in comparison with the performance of the DEFT'07 participants

- decision rule 4.3;
- Artificial Neural Network (ANN) with error back propagation learning.

Tables 5.23 and 5.26 present the F-scores obtained on the test corpora using the semi-supervised dimensionality reduction method based on the term clustering. The best values are shown in bold. Results of the k-NN algorithm are presented with the best k parameter. F-scores obtained on the test corpus are considered as classification quality measures.

The experimental results obtained on the T1 corpus using the reduced feature space are shown in Table 5.23, where the dimensionality reduction of the feature space has been performed by the feature extraction method based on the semi-supervised term clustering with 100 term clusters for each category. The novel TW combined with the Rocchio algorithm provides the best classification quality ($Fscore = 0.837$), which outperforms the average result of the DEFT 2008 participants (average $Fscore = 0.826$, the best $Fscore = 0.894$). The obtained performance is, however, worse than the result on the initial feature space ($Fscore = 0.876$) and the feature space reduced using the feature selection method ($Fscore = 0.857$). According to the results presented in Table 5.23 the best performance is achieved with either the novel TW or the tf.idf 3 ($\gamma = 0.5$) text preprocessing techniques.

Table 5.24 presents the performance obtained on the T1 corpus using the reduced term set, where the feature space has been reduced using the semi-supervised feature extraction approach (50 term clusters per category). The best classification quality ($Fscore = 0.841$) is provided with the novel TW as a text preprocessing technique and the Rocchio algorithm as a classification method. This result outperforms the average F-score (average $Fscore = 0.826$,the best $Fscore = 0.894$) obtained by the participants of the DEFT 2008 and it also outperforms the result produced by the semi-supervised dimensionality reduction algorithm with 100 term clusters per category ($Fscore = 0.837$). Table 5.24 shows that using the novel TW outperforms other text preprocessing methods.

Figure 5.7 shows that the performance of the classification algorithms obtained on the reduced term sets of the corpus T1 improves in 3 cases (tf.idf 3 $\gamma = 0.9$, tf.idf 4 $\gamma = 0.5; 0.9$) and becomes worse in 8 cases (novel TW, ConfWeight, tf.idf 1, tf.idf 2, tf.idf 3 $\gamma = 0.1; 0.5$ and tf.idf 4 $\gamma = 0.1; 0.5$). The use of 50 term clusters per category improves the classification quality using the tf.idf 4 $\gamma = 0.1$ text preprocessing, in 7 cases (ConfWeight, tf.idf 1, tf.idf 3 $\gamma = 0.1; 0.5; 0.9$, tf.idf 4$\gamma = 0.5; 0.9$) produces worse representations than the use of 100 term clusters per category and in 2 cases (novel TW and tf.idf 2) there was no significant change in the performance. However, it should be mentioned that the best performance on the reduced term sets have been achieved using the novel TW, and on the initial term set - with the novel TW and ConfWeight.

Table 5.25 presents the results of the numerical experiments conducted on the T2 corpus using the semi-supervised dimensionality reduction method with 100 term clusters for each

	k-NN	Bayes	SVM Linear	Rocchio	Dec. rule 4.3	ANN
Novel TW	**0,825 (11)**	**0,692**	0,789	**0,837**	**0,830**	0,774
Conf Weight	0,710 (8)	0,576	0,667	0,723	0,830	0,691
tf.idf 1	0,816 (15)	0,613	0,803	0,809	0,806	0,810
tf.idf 2	0,803 (6)	0,597	0,798	0,806	0,800	0,806
tf.idf 3 $\gamma = 0.1$	0,806 (5)	0,622	0,804	0,820	0,807	0,814
tf.idf 3 $\gamma = 0.5$	0,805 (11)	0,660	**0,808**	0,815	0,812	**0,817**
tf.idf 3 $\gamma = 0.9$	0,799 (14)	0,654	0,797	0,809	0,802	0,799
tf.idf 4 $\gamma = 0.1$	0,803 (15)	0,620	0,799	0,814	0,807	0,814
tf.idf 4 $\gamma = 0.5$	0,805 (14)	0,645	0,799	0,810	0,803	0,821
tf.idf 4 $\gamma = 0.9$	0,792 (4)	0,661	0,787	0,802	0,794	0,794

Table 5.23: F-scores obtained using the semi-supervised dimensionality reduction method (100 term clusters per category): T1

	k-NN	Bayes	SVM Linear	Rocchio	Dec. rule 4.3	ANN
Novel TW	**0,830 (15)**	**0,714**	**0,823**	**0,841**	**0,833**	**0,835**
Conf Weight	0,338 (14)	0,283	0,250	0,319	0,815	0,246
tf.idf 1	0,801 (9)	0,618	0,796	0,807	0,800	0,798
tf.idf 2	0,790 (15)	0,628	0,792	0,798	0,795	0,805
tf.idf 3 $\gamma = 0.1$	0,796 (13)	0,633	0,794	0,803	0,799	0,809
tf.idf 3 $\gamma = 0.5$	0,793 (13)	0,669	0,789	0,800	0,792	0,807
tf.idf 3 $\gamma = 0.9$	0,789 (14)	0,619	0,784	0,794	0,725	0,791
tf.idf 4 $\gamma = 0.1$	0,792 (14)	0,597	0,797	0,804	0,800	0,819
tf.idf 4 $\gamma = 0.5$	0,786 (15)	0,624	0,783	0,791	0,790	0,789
tf.idf 4 $\gamma = 0.9$	0,780 (15)	0,622	0,783	0,782	0,778	0,782

Table 5.24: F-scores obtained using the semi-supervised dimensionality reduction method (50 term clusters per category): T1

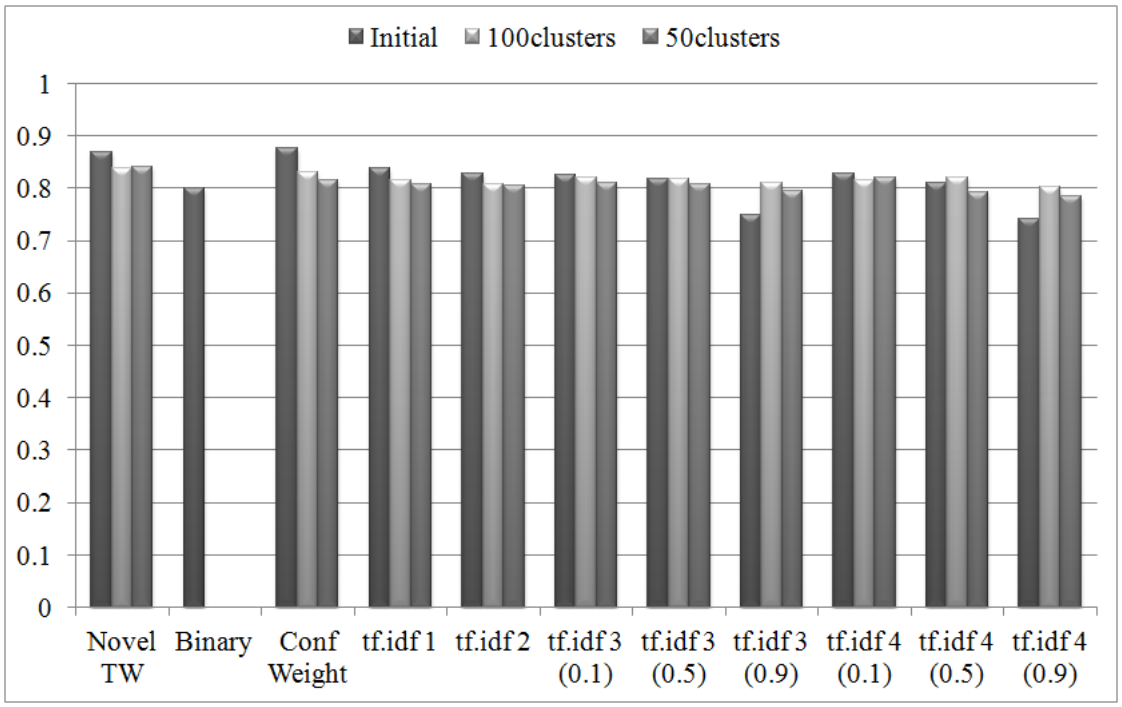

Figure 5.7: The best algorithmic performance obtained on the T1 corpus using different text preprocessings

category. The highest F-score ($Fscore = 0.843$) has been obtained using the novel TW as a text preprocessing technique and the k-NN ($k = 13$) as a classification method. This combination of methods outperforms the average F-score achieved by the DEFT 2008 participants (average $Fscore = 0.811$,the best $Fscore = 0.880$). However, it is worse than the result on the initial feature space ($Fscore = 0.852$) and the performance on the reduced feature space obtained using the term filtering approach ($Fscore = 0.858$). Table 5.25 shows that the novel TW produces higher F-scores than other preprocessing techniques.

Using 50 term clusters for each category the best performance of classification algorithms in combination with several semi-supervised text preprocessing techniques ($Fscore = 0.842$) has been achieved by the novel TW as a term weighting algorithm and the k-NN ($k = 15$) as a classification method. This result outperforms the average F-score (average $Fscore = 0.811$,the best $Fscore = 0.880$) obtained by the participants of the DEFT 2008 and it is similar to the performance produced by 100 term clusters per category ($Fscore = 0.843$). Table 5.26 shows that using the novel TW all classification algorithms perform better, excluding the ANN where the best result has been obtained with the tf.idf 1 preprocessing.

Figure 5.8 compares the best performances obtained using different text preprocessing algorithms with the initial and reduced (100 and 50 term clusters per category) term sets on the corpus T2. It has been shown that the performance of the classification algorithms

	k-NN	Bayes	SVM Linear	SVM ML	Rocchio	Dec. rule 4.3	ANN
Novel TW	**0,843 (13)**	**0,687**	**0,842**	**0,839**	**0,837**	**0,837**	**0,836**
Conf Weight	0,819 (14)	0,609	0,815	0,811	0,806	0,831	0,813
tf.idf 1	0,839 (14)	0,632	0,840	0,836	0,835	0,835	0,834
tf.idf 2	0,834 (12)	0,644	0,835	0,832	0,830	0,829	0,828
tf.idf 3 $\gamma = 0.1$	0,836 (13)	0,632	0,834	0,833	0,831	0,835	0,832
tf.idf 3 $\gamma = 0.5$	0,835 (12)	0,677	0,834	0,833	0,830	0,832	0,827
tf.idf 3 $\gamma = 0.9$	0,831 (11)	0,681	0,830	0,826	0,826	0,830	0,821
tf.idf 4 $\gamma = 0.1$	0,835 (12)	0,630	0,832	0,831	0,828	0,829	0,815
tf.idf 4 $\gamma = 0.5$	0,833 (15)	0,660	0,833	0,829	0,829	0,830	0,820
tf.idf 4 $\gamma = 0.9$	0,832 (15)	**0,687**	0,830	0,828	0,830	0,829	0,815

Table 5.25: F-scores obtained using the semi-supervised dimensionality reduction method (100 term clusters per category): T2

	k-NN	Bayes	SVM Linear	SVM ML	Rocchio	Dec. rule 4.3	ANN
Novel TW	**0,842 (15)**	**0,696**	**0,841**	**0,838**	**0,836**	**0,838**	0,802
Conf Weight	0,819 (15)	0,638	0,817	0,814	0,809	0,832	0,815
tf.idf 1	0,840 (15)	0,670	0,837	0,836	0,834	0,835	**0,832**
tf.idf 2	0,838 (15)	0,659	0,835	0,832	0,831	0,833	0,813
tf.idf 3 $\gamma = 0.1$	0,834 (12)	0,680	0,835	0,833	0,832	0,833	0,825
tf.idf 3 $\gamma = 0.5$	0,835 (14)	0,691	0,831	0,830	0,830	0,831	0,821
tf.idf 3 $\gamma = 0.9$	0,830 (15)	0,685	0,829	0,826	0,827	0,827	0,790
tf.idf 4 $\gamma = 0.1$	0,840 (4)	0,651	0,834	0,833	0,828	0,831	0,819
tf.idf 4 $\gamma = 0.5$	0,831 (15)	0,680	0,827	0,828	0,825	0,828	0,816
tf.idf 4 $\gamma = 0.9$	0,823 (15)	0,665	0,822	0,822	0,821	0,822	0,805

Table 5.26: F-scores obtained using the semi-supervised dimensionality reduction method (50 term clusters per category): T2

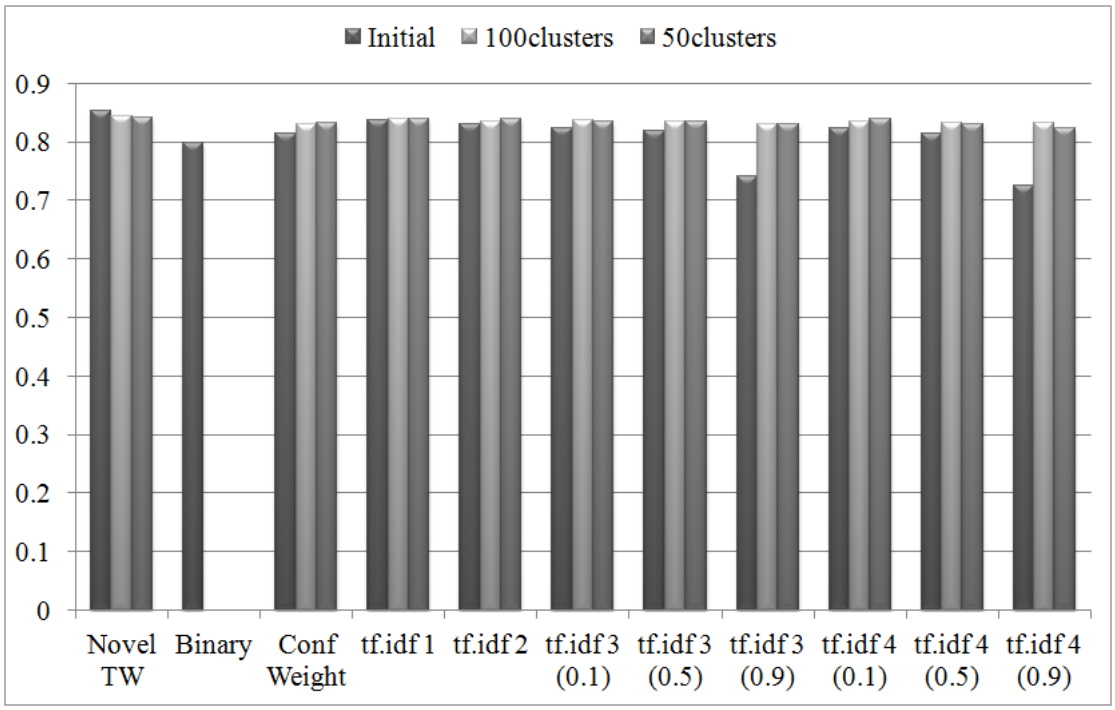

Figure 5.8: The best algorithmic performance obtained on the T2 corpus using different text preprocessing methods

obtained on the reduced term sets improves with all text representations excluding the novel TW and in case of the tf.idf 1 text preprocessing the performance has not changed significantly. The use of 50 term clusters per category in 4 cases (novel TW, tf.idf 3 $\gamma = 0.1$, tf.idf 4$\gamma = 0.5; 0.9$) produces worse representations than the use of 100 term clusters per category; in 3 cases (ConfWeight, tf.idf 2, tf.idf 4 $\gamma = 0.1$) it improves the results and in 3 cases (tf.idf 1, tf.idf 3 $\gamma = 0.5; 0.9$) it did not have an impact. However, it should be mentioned, that despite of the fact, that the results obtained on the reduced feature space using the novel TW are worse than the results on the initial feature space, algorithms still perform better using the novel TW than other text representations.

The comparison of the best combinations of the proposed methods obtained using the initial and reduced (100 and 50 term clusters per category) term sets with the results of the participants of DEFT 2008 is shown in Table 5.27. The best performance has been obtained by Torres-Moreno team for the T2 corpus ($Fscore = 0.880$) and by Buffoni team for the T1 corpus ($Fscore = 0.894$). The best combinations introduced in this thesis have reached 0.8755 and 0.852 for the T1 and T2 corpora using the initial feature space, which takes 3rd and 5th places in the common table.

Team	T1	T2	Rank
J. M. Torres-Moreno (LIA)	0.883 (2)	0.880 (1)	1
M. Plantié (LGI2P/LIRMM)	0.853 (6)	0.858 (4)	5
E. Charton (LIA)	0.875 (5)	0.879 (2)	3
D. Buffoni (LIP6)	0.894 (1)	0.876 (3)	2
G. Cleuziou (LIFO/INaLCO)	0.790 (10)	0.821 (9)	7
F. Rioult (GREYC)	0.849 (7)	0.838 (8)	6
Initial term set	**0.876 (3)**	**0.852 (5)**	**4**
Reduced term set using feature selection	**0.857 (4)**	**0.858 (4)**	**4**
Reduced term set using feature extraction (100 term clusters for each category)	**0.837 (9)**	**0.843 (6)**	**6**
Reduced term set using feature extraction (50 term clusters for each category)	**0.841 (8)**	**0.842 (7)**	**6**
Average result obtained by participants	*0.826*	*0.811*	

Table 5.27: Total results on the corpora DEFT 2008 in comparison with the performance of the DEFT'08 participants

5.4 Summary

This chapter reports on text classification experiments conducted on six different corpora (topic categorization: 2 datasets of document categorization, 1 dataset of natural language call routing; opinion mining: 3 datasets) using several classification methods with different text preprocessing. The aim of these experiments is a comparison of a proposed text preprocessing scheme including a novel term weighting algorithm and a novel dimensionality reduction method with the state-of-the-art text preprocessing algorithms in terms of the classification quality and the required CPU computational time. This chapter also provides a comparison of a metaheuristic optimization method used for SVM and ANN-based text classifiers generation with the state-of-the-art classification algorithms

Eight TF-IDF modifications (TF-IDF 1, TF-IDF 2, TF-IDF 3 with $\alpha = 0.1, 0.5, 0.9$ and TF-IDF 4 with $\alpha = 0.1, 0.5, 0.9$), ConfWeight and novel term weighting approach have been used as text preprocessing techniques, where TF-IDF variations are unsupervised term weighting methods and ConfWeight and novel TW are supervised term weighting algorithms. k-nearest neighbors algorithm with $1 \leq k \leq 15$, Bayes classifier, Support Vector Machine (SVM) with different kernel functions, SVM generated using a metaheuristic called COBRA (Co-Operation of Biology Related Algorithms), Rocchio classifier, artificial neural network trained using back propagation and ANN-based classifier generated and trained with COBRA have been applied as classification algorithms.

First, the numerical results on all preprocessings have shown that the proposed term weighting is competitive with the existing models. It provides similar results as ConfWeight and outperforms TF-IDF modifications.

Numerical experiments which measured the required computational time have shown that despite the fact that ConfWeight method is more effective for classification than all TF-IDF modifications; ConfWeight consumes more computational time since it requires the Student's law calculation, Wilson proportion and confidence intervals. The novel term weighting method gives similar or better classification quality than ConfWeight but it requires the same amount of time as all TF-IDF modifications (in 3-4 times less computational time than ConfWeight preprocessing).

The algorithm performances after dimensionality reduction (term clustering) with 100 and 50 term clusters per category have been provided.

It is shown that the proposed term clustering with the cooperative coevolutionary algorithm reduces the feature space dimensionality from hundreds of thousands words to 500 term clusters and this reduction does not affect significantly on the performance of the classification algorithms.

Numerical experiments conducted in this chapter have shown computational and classification efficiency of the proposed method (the novel TW) in comparison with existing text

preprocessing techniques. Although the participants of the DEFT 2007 and 2008 have used stop word filtering and stemming, the fully statistical approaches developed in this thesis can compete and in some cases outperform the text classifiers proposed by the participants. The dimensionality reduction algorithms introduced in this thesis reduce significantly the feature space (T2 corpus: from about 250000 to 200 terms) with only a small loss in the performance (T2 corpus: from 0.852 to 0.841 F-score). The automatically generated and optimized SVM- and ANN-based classifiers have shown the best performance in comparison with the standard state-of-the-art classification methods.

Chapter 6

Conclusions and Future Directions

6.1 Summary of Contributions

In this thesis, novel machine learning algorithms have been developed in the fields of text classification and text preprocessing. Text preprocessing methods proposed in this thesis include a novel term weighting method and a novel feature space dimensionality reduction technique. This thesis contributes to the field of text classification in two main parts. The first one is an unsupervised technique that improves the classification performance on non-uniform categories. The second part includes a metaheuristic optimization algorithm for generation and adjustment of the popular classification algorithms: Support Vector Machine and Artificial Neural Network applied to the document classification task.

In order to estimate the performance of the proposed text preprocessing techniques and text classification algorithms six corpora (DEFT 2007: Books, Games and Debates; DEFT 2008: T1 and T2; SpeechCycle) have been used. These corpora differ from each other in language (French and English), task (opinion mining, topic categorization and natural language call routing), document length (1-15000 terms) and dimension of the feature space (3500-250000 terms). The corpora also differ in the way of obtaining these corpora: while most of the databases contain articles with the grammatically correct sentences taken from the newspaper "Le Monde" and Wikipedia, the natural language call routing corpus (SpeechCycle) has been created by automatic speech recognition (ASR) engine from the acoustic signal of the real callers.

The novel term weighting technique that contributes to the field of text preprocessing has been introduced in the first part of Chapter 4. It is a supervised term weighting algorithm that is based on a modified formula adopted from the fuzzy rules membership calculation. This heuristic allows classification algorithms to achieve good performance in relatively short time, since the problem of many text preprocessing methods is a high CPU time requirement. In this thesis a set of different classifiers commonly used for text classification are

applied in order to test the quality of the proposed term weighting algorithm. Due to the fact that some text preprocessing methods perform significantly better in combination with the particular classifiers, it is important to show how the chosen preprocessing technique and the given classification algorithm influence each other in terms of the obtained performance. The variety of classifiers perform with the novel term weighting method better or almost the same in comparison with eight modifications of TF-IDF preprocessings, binary representation and ConfWeight preprocessing on six corpora for text classification (opinion mining: 3 data sets; topic categorization: 2 data sets; natural language call routing: 1 data set). The computational time for different preprocessing methods has been compared (each preprocessing algorithm has been run 10 times). Numerical experiments conducted in this thesis have shown that the proposed term weighting approach provides better (than TF-IDF modifications and binary representation) or nearly the same (as ConfWeight) classification quality, however it requires 3-4 times less computational time (than ConfWeight) depending on the task.

The emerged (own) publications related to this thesis contribution are Gasanova *et al.* (2013c,a, 2014b); Sergienko *et al.* (2013, 2014), .

In the second part of Chapter 4 a semi-supervised method for the feature space dimensionality reduction has been proposed. This text preprocessing method is based on a combination with unsupervised learning algorithms that reduce the feature space dimension without significant loss in performance. The introduced method consists of the term set reduction and application of optimization algorithm in order to adjust the common weights of the obtained term clusters.

First, the term set size has been reduced by a hierarchical agglomerative clustering applied to the feature space (term clustering). This clustering is based on the weights obtained by term weighting algorithms. It detects synonyms or terms with similar weights in the category. This algorithm can be performed in a combination with different term weighting techniques. After the initial terms are transformed into term clusters, the common weights for all term clusters are counted and a new data representation is recalculated. Since the reduced term set is relatively small, it can be useful to apply some optimization methods in order to improve text representation.

Second, since there is no evidence that the common weights of the term clusters lead even to local optima of the objective function yet alone to the global optimum, the optimization algorithm is applied to adjust the weights of the obtained term clusters. The objective function of the classification quality depending on the term cluster weights is not analytically defined, therefore, heuristic algorithms can be applied, since they do not require such information about the behaviour of the objective function. The cooperative scheme of coevolutionary genetic algorithm has been proposed as an optimization tool since the task to find optimal weights of the obtained term clusters can be naturally separated into the subtasks (subpop-

ulations in the coevolutionary algorithm). Each subpopulation is considered as a category of the classification task, it evolves during several generations without sharing the information between subpopulations, then the evolution stops and algorithms updates the best obtained weights by collecting information from all subpopulations, then the cycle goes on. In order to provide the local convergence the local search algorithm is applied (several times the random bit of the chromosome is inverted, the fitness function is recalculated and if the change makes profit than it is kept) only to the best obtained individual on each generation. Since this hybridization is performed only to the one individual on every generation, it does not significantly slow the algorithm down.

Furthermore, in order to overcome the overfitting problem on some of our corpora (Books and Games) that all applied algorithms meet, the fitness function of the genetic algorithm (the optimization criterion) has been modified. If the accuracy on the training set reaches almost 100%, the classification quality (accuracy and F-score) on the test set can still improve. The idea is to combine the existing supervised criterion (accuracy obtained on the training set) with the unsupervised learning in order to obtain better partition into classes in terms of cluster analysis. In this work the maximization of the inter-cluster similarity (the average distance between centroids of the obtained clusters) has been used as an additional criterion. This unsupervised criterion is added only to candidate solutions that have already reached over 0.95 of accuracy rate on the training set. This change of the objective function leads to the improvement of the performance of the classification algorithms using this document representation.

This algorithm has been designed for the corpora with the large term set and grammatically correct sentences, where each document contains many terms. In this thesis DEFT 2007/2008 corpora satisfy these criteria and SpeechCycle corpus has its own peculiarities that make the application of this approach pointless. The main reason is that the SpeechCycle document (call) consists of only a few words, each of them makes a great impact on the choice of the category (key words).

All methods described in this thesis do not use any information specific for the domain or language, i.e. no stop word or ignore word lists have been used to filter the term set.

The emerged publications related to this thesis contribution are Gasanova *et al.* (2013b, 2014a).

Chapter 4 also describes the contributions to the field of text classification. First, a novel semi-supervised approach has been introduced. It aims to improve the performance of the classification methods in the corpora with non-uniform classes. Non-uniform categories contain elements from different sources (such as SpeechCycle corpus with the non-uniform class TE_NOMATCH) and these classes are relatively common in real-world applications. The proposed method is based on the assumption that if such non-uniform class is divided into

the set of well-defined subclasses using unsupervised learning algorithm, then the performance of the classification method will be improved. In case of SpeechCycle corpus calls can be routed to one of the previous categories or to one of the obtained subclasses of the class _TE_NOMATCH. It should be noted that in this case the classification accuracy obtained on the whole training set measures the clustering quality obtained on a part of the training set. In this work genetic algorithm with integers, learning vector quantization algorithm optimized with genetic algorithm and hierarchical agglomerative clustering method with different linkage criteria including single-linkage, complete-linkage, average-linkage and Ward's criterion have been applied. The biggest improvement has been achieved using the hierarchical agglomerative clustering with the Hamming metric and the Ward criterion.

The emerged publications related to this thesis contribution are Gasanova *et al.* (2013a,c).

Chapter 4 provides the description of the novel approach for Support Vector Machine (SVM) and Artificial Neural Network (ANN) generation and parameters optimization. The obtained classifiers perform better than the standard SVM and ANN. The cooperation of biology related algorithms (COBRA) has been used to generate SVM and ANN. In this thesis the original version, constrained and binary modifications have been described. For generating the SVM-machine the original COBRA is used: each individual in all populations represents a set of kernel function's parameters. Then for each individual the constrained modification of COBRA is applied for finding vector w and the shift factor b. And finally, the individual with the best classification rate is chosen as the designed classifier. The binary modification of COBRA has been used for the classifier structure optimization and the original COBRA has been used for the weight coefficients adjustment both within structure selection process and for the final tuning of the best selected structure.

The emerged publication related to this thesis contribution is Akhmedova *et al.* (2014).

The details of algorithm implementation and experimental setup are discussed in Chapter 5. The performances of classification algorithms on the six different corpora using several standard text preprocessings and the proposed term weighting technique have been provided. As a classification methods k-nearest neighbors algorithm with $1 \leq k \leq 15$, Bayes classifier, Support Vector Machine with different kernel functions, SVM generated using COBRA (Co-Operation of Biology Related Algorithms), Rocchio classifier, artificial neural network and the decision rule which has been used in optimization algorithm as a base for an objective function.

6.2 Conclusions

The main conclusions that can be drawn from the approaches described in the previous section are as follows.

Chapter 4 shows that the classification performance can be improved by clustering the non-uniform class into the set of well-defined subclusters.

The term weighting approach also proposed in Chapter 4 gives similar results as ConfWeight preprocessing and significantly better than TF-IDF modifications which have been used for comparison. The novel TW shows better performance in terms of computational time since it does not require the calculations of the confidence intervals and other functions which ConfWeight uses.

The method applied on SpeechCycle corpus to divide the largest class of non-uniform calls has provided significant improvement of the classification accuracy. The problem related to this class lies in its non-uniform structure, it includes utterances that are very different from each other: calls that have no meaning, calls that can be routed to more than one informative class and calls that cannot be routed to any of the existing categories. The idea of the introduced approach is a combination of the supervised and unsupervised learning models. The clustering algorithm divides the non-uniform class into the set of subclasses (an unsupervised part) and, then, the classification process is applied to estimate the quality of the obtained clustering (a supervised part). This classification process assigns for the given object one of the existing informative categories or one of the new subclasses. Different clustering methods have been compared including a genetic algorithm with integers, vector quantization networks trained by a genetic algorithm, hierarchical agglomerative clustering with different metrics (the metrics are single-linkage, complete-linkage, average linkage and Ward's criterion). The best result has been provided by the hierarchical agglomerative clustering with the Hamming metric and the Ward's criterion.

In Chapter 5 the simulation results using different preprocessings and classification methods have been presented. It has been shown that the generated Support Vector Machine and Artificial Neural Network proposed in Chapter 4 outperforms the results of standard SVM and ANN on the given corpora with all preprocessings as well as other state-of-the-art text classifiers.

Finally, the results of the algorithms performance on preprocessings with reduced term set (two variants: 50 and 100 term clusters for each of the categories) have been presented in Chapter 5. The comparison between the results obtained on the initial term set and the reduced one shows that using the reduced term set without optimization of the term cluster weights produces significantly worse performance of the classification methods.

The performances of the preprocessings with optimal weights of term clusters (on both 50 and 100 term clusters) have been also presented in Chapter 5. It has been shown that the optimization of the weights of term clusters improves classification results using all preprocessings and all classification methods.

All proposed methods for text preprocessing and dimensionality reduction which have

been described in this thesis are cross lingual and domain independent. They are based on the Bag-of-Words model and use statistics of the word occurrence in the documents from different categories. The introduced approach takes advantage of both supervised learning model and unsupervised one in order to improve the performance of the classification algorithm. All methods have been evaluated using six different corpora for text classification and compared with the state-of-the-art text classification approaches combined with standard text preprocessing techniques. The developed algorithms have been also compared with the methods designed by participants of the DEFT 2007/2008 campaigns. It has been shown that the proposed algorithms can compete or in some cases outperform the existing text classification and information retrieval systems in spite of these systems use linguistic and domain related information which makes them specific for the given task.

6.3 Future Directions

The rapid growth of textual data in digital form greatly increases a need for powerful text mining algorithms. Text data mining is an interdisciplinary field and since it connects many research communities such as data mining, natural language processing, information retrieval and machine learning with applications in many different areas, text preprocessing and text classification have recently attracted a lot of attention.

Understanding semantics of text data is vital to text mining. The current approaches to text preprocessing mostly rely on bag-of-words representation, however, it is clearly desirable to go beyond such a simple representation, because bag-of-words model excludes a lot of important information. Semantic representation provided by information extraction methods relies on supervised learning and usually works well when enough training data is available, which clearly restricts its applications. Therefore, it is important to develop text preprocessing and text classification methods that can scale to multiple domains.

Since many text classification algorithms rely on supervised learning, their effectiveness highly depends on the amount of training data available. It is generally expensive to create large amounts of training data exclusively for each particular task. Domain adaptation and transfer learning methods can alleviate this problem by using training data from a related domain or a related task. However, existing approaches still have many limitations and are generally not effective when there is no or little training data in a target domain.

Since context information such as authors, sources, time or more complicated information networks is generally associated with text data, in many applications it is important to consider the context and also user preferences in text mining. Therefore, it is important to further incorporate this context data to improve text classification.

In many applications of text classification the amount of text data is huge and it is

usually increasing over time. Therefore it becomes difficult or impossible to store the data in one machine, which creates a need to develop parallel algorithms for text classification. Parallel unsupervised and supervised methods can run on a cluster of computers to perform text mining tasks in parallel. This future direction is clearly related to cloud computing and data-intensive computing fields.

With the spread of web-based and other information retrieval methods to other corpora and tasks, it has become particularly useful to apply text classification algorithms to different languages and to use the knowledge obtained in corpora in one language to another. For cross-language text mining it is useful to cluster a document collection, where similar semantic topics written in different languages are placed in the same cluster. Such cross-lingual application can be used to transfer knowledge between different data domains.

One of the applications of the algorithms developed in this thesis lies in the field of sentiment analysis and opinion mining. This domain has been a challenging research area in recent years due to many practical applications and it has become the topic of the management science as well as the computer science. A considerable amount of text on web pages are product reviews and opinions of different users. Mining this kind of text data to reveal and summarize the opinions about the topic has many commercial applications. Since every company wants to monitor how clients evaluate its service in comparison with the service of its competitors, the field of opinion mining will be kept vibrant for the future.

Natural language understanding remains an open problem and the interest in this field grows rapidly, since the large number of commercial applications include or would like to include the possibility to operate using the speech.

The topic categorization task discussed in the thesis is the oldest of the three problems and most of the techniques now applied for natural language understanding and opinion mining have been first designed for topic categorization.

There are many other applications of the automatic text classification including spam filtering and e-mail routing that attempt to save the time for the people. Language identification and genre classification are also important for the search engines. The readability assessment becomes a part of the task performed by the automatic search systems in order to find suitable material for the given type and age of the reader.

In relation to the text preprocessing techniques discussed in Chapter 2.1 future work can be directed to the further combinations of unsupervised and supervised methods. Unsupervised hierarchical models such as deep learning which uses the labels only on the last level of the text processing. The deep learning methods now play important role in text processing algorithms. The deep learning technique has been introduced in Hinton & Salakhutdinov (2006) and it is based on the artificial neural net model where only last layer is trained with class labels and other layers use unsupervised approach. It allows these methods to form their own features

without human supervision. This idea has been applied in various tasks and have shown promising results.

With regard to the improvement of the SVM-based text classifier, the direct step from the algorithm developed in this thesis will be the design of SVM-classifiers using other types of kernel functions (for example, sigmoid functions) with automated choice of the best of them.

Directions of the future research concerning the ANN generation are heterogeneous: the development of modifications for constrained and multicriteria optimization, the improvement of the cooperation and competition scheme within the approach, addition of other algorithms in cooperation and invention of the algorithms selection techniques, development of modification for mixed optimization problems, etc.

The combinations of the supervised and unsupervised models will be further developed. An interesting idea is to add unsupervised criterion to the objective function. The importance of the supervised and unsupervised part can be weighted dynamically depending on the optimization stage of the classification algorithm.

Bibliography

2007. *Proceedings of the Third DEFT Workshop.*

2008. *Proceedings of the 4th DEFT Workshop.*

Aggarwal, Charu C, & Zhai, ChengXiang. 2012. *Mining text data.* Springer.

Akhmedova, S., Semenkin, E., Gasanova, T., & Minker, W. 2014. Co-operation of biology related algorithms for support vector machine automated design. *International Conference on Engineering and Applied Sciences Optimization (OPT-i).*

Akhmedova, Sh., & Semenkin, E. 2014. Co-Operation of Biology Related Algorithms Meta-Heuristic in ANN-Based Classifiers Design. *In: IEEE World Congress on Computational Intelligence (IEEE WCCI).*

Akhmedova, Shakhnaz, & Semenkin, Eugene. 2013. Co-Operation of Biology Related Algorithms. *Pages 2207–2214 of: Evolutionary Computation (CEC), 2013 IEEE Congress on.* IEEE.

Albalate, Amparo, Rhinow, Steffen, & Suendermann, David. 2010. A Non-parameterised Hierarchical Pole-based Clustering Algorithm (HPoBC). *Pages 350–356 of: ICAART (1).*

Ando, Rie Kubota, & Zhang, Tong. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, **6**, 1817–1853.

Baker, James Edward. 1985. Adaptive selection methods for genetic algorithms. *Pages 101–111 of: Proceedings of the international conference on genetic algorithms and their applications.*

Baker, L Douglas, & McCallum, Andrew Kachites. 1998. Distributional clustering of words for text classification. *Pages 96–103 of: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

Balcan, Maria-Florina, Blum, Avrim, & Yang, Ke. 2004. Co-Training and Expansion: Towards Bridging Theory and Practice. *Pages 89–96 of: NIPS*, vol. 17.

Batal, Iyad, & Hauskrecht, Milos. 2009. Boosting KNN text classification accuracy by using supervised term weighting schemes. *Pages 2041–2044 of: Proceedings of the 18th ACM conference on Information and knowledge management.* ACM.

Berkhin, Pavel. 2006. A survey of clustering data mining techniques. *Pages 25–71 of: Grouping multidimensional data.* Springer.

Blanchard, Antoine. 2007. Understanding and customizing stopword lists for enhanced patent mapping. *World Patent Information*, **29**(4), 308–316.

Blei, David M, Ng, Andrew Y, & Jordan, Michael I. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, **3**, 993–1022.

Blum, Avrim, & Mitchell, Tom. 1998. Combining labeled and unlabeled data with co-training. *Pages 92–100 of: Proceedings of the eleventh annual conference on Computational learning theory.* ACM.

Blum, Avrim L, & Langley, Pat. 1997. Selection of relevant features and examples in machine learning. *Artificial intelligence*, **97**(1), 245–271.

Boser, Bernhard E, Guyon, Isabelle M, & Vapnik, Vladimir N. 1992. A training algorithm for optimal margin classifiers. *Pages 144–152 of: Proceedings of the fifth annual workshop on Computational learning theory.* ACM.

Brown, Peter F, Desouza, Peter V, Mercer, Robert L, Pietra, Vincent J Della, & Lai, Jenifer C. 1992. Class-based n-gram models of natural language. *Computational linguistics*, **18**(4), 467–479.

Cannas, Laura Maria, Dessì, Nicoletta, & Dessì, Stefania. 2012. A Model for Term Selection in Text Categorization Problems. *Pages 169–173 of: Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on.* IEEE.

Carvalho, Vitor R, & Cohen, William W. 2005. On the collective classification of email speech acts. *Pages 345–352 of: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

Chakrabarti, Soumen, Dom, Byron, Agrawal, Rakesh, & Raghavan, Prabhakar. 1997. Using taxonomy, discriminants, and signatures for navigating in text databases. *Pages 446–455 of: VLDB*, vol. 97.

Chawla, Nitesh V, & Karakoulas, Grigoris I. 2005. Learning From Labeled And Unlabeled Data: An Empirical Study Across Techniques And Domains. *J. Artif. Intell. Res.(JAIR)*, **23**, 331–366.

Chen, Jingnian, Huang, Houkuan, Tian, Shengfeng, & Qu, Youli. 2009. Feature selection for text classification with Naïve Bayes. *Expert Systems with Applications*, **36**(3), 5432–5435.

Christopher, D Manning, Prabhakar, Raghavan, & Hinrich, Schütze. 2008. Introduction to information retrieval. *An Introduction To Information Retrieval*, 151–177.

Chu-Carroll, Jennifer, & Carpenter, Bob. 1999. Vector-based natural language call routing. *Computational linguistics*, **25**(3), 361–388.

Clack, Chris, Farringdon, Johnny, Lidwell, Peter, & Yu, Tina. 1997. Autonomous document classification for business. *Pages 201–208 of: Proceedings of the first international conference on Autonomous agents*. ACM.

Cleuziou, Guillaume, Martin, Lionel, & Vrain, Christel. 2004. Poboc: an overlapping clustering algorithm. *Application to rule-based classification and textual data*, 440–444.

Cohen, William W. 1996. Learning rules that classify e-mail. *Page 25 of: AAAI Spring Symposium on Machine Learning in Information Access*, vol. 18. California.

Colace, Francesco, De Santo, Massimo, Greco, Luca, & Napoletano, Paolo. 2013. Improving text retrieval accuracy by using a minimal relevance feedback. *Pages 126–140 of: Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Springer.

Colace, Francesco, De Santo, Massimo, Greco, Luca, & Napoletano, Paolo. 2014. Text classification using a few labeled examples. *Computers in Human Behavior*, **30**, 689–697.

Collobert, Ronan, & Weston, Jason. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *Pages 160–167 of: Proceedings of the 25th international conference on Machine learning*. ACM.

Cortes, Corinna, & Vapnik, Vladimir. 1995. Support-vector networks. *Machine learning*, **20**(3), 273–297.

Deb, Kalyanmoy. 2000. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, **186**(2), 311–338.

Debole, F., & Sebastiani, F. 2003. Supervised Term Weighting for Automated Text Categorization. *In: Proc. ACM Symp. Applied Computing*.

Deerwester, Scott C., Dumais, Susan T, Landauer, Thomas K., Furnas, George W., & Harshman, Richard A. 1990. Indexing by latent semantic analysis. *JASIS*, **41**(6), 391–407.

Deisy, C, Gowri, M, Baskar, S, Kalaiarasi, SMA, & Ramraj, N. 2010. A novel term weighting scheme MIDF for text categorization. *Journal of Engineering Science and Technology*, **5**(1), 94–107.

Demiriz, Ayhan, Bennett, Kristin P, & Embrechts, Mark J. 1999. Semi-supervised clustering using genetic algorithms. *Artificial neural networks in engineering (ANNIE-99)*, 809–814.

Dietterich, Thomas G. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, **10**(7), 1895–1923.

Dong, Hai, Hussain, Farookh Khadeer, & Chang, Elizabeth. 2011. A framework for discovering and classifying ubiquitous services in digital health ecosystems. *Journal of Computer and System Sciences*, **77**(4), 687–704.

Dragut, Eduard, Fang, Fang, Sistla, Prasad, Yu, Clement, & Meng, Weiyi. 2009. Stop word and related problems in web interface integration. *Proceedings of the VLDB Endowment*, **2**(1), 349–360.

Dumais, Susan, Platt, John, Heckerman, David, & Sahami, Mehran. 1998. Inductive learning algorithms and representations for text categorization. *Pages 148–155 of: Proceedings of the seventh international conference on Information and knowledge management*. ACM.

E.-H. Han, G. Karypis, & Kumar, V. 2001. Text Categorization Using Weight Adjusted K-Nearest Neighbor Classification. *In: Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining*.

Eiben, Agoston E, & Smith, James E. 2003. *Introduction to evolutionary computing*. springer.

Eirinaki, Magdalini, Pisal, Shamita, & Singh, Japinder. 2012. Feature-based opinion mining and ranking. *Journal of Computer and System Sciences*, **78**(4), 1175–1184.

Evanini, Keelan, Suendermann, David, & Pieraccini, Roberto. 2007. Call classification for automated troubleshooting on large corpora. *Pages 207–212 of: Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE.

Feng, Wei, Xie, Lei, & Liu, Zhi-Qiang. 2010. Multicue graph mincut for image segmentation. *Pages 707–717 of: Computer Vision–ACCV 2009*. Springer.

Figueiredo, Fábio, Rocha, Leonardo, Couto, Thierson, Salles, Thiago, Gonçalves, Marcos André, & Meira Jr, Wagner. 2011. Word co-occurrence features for text classification. *Information Systems*, **36**(5), 843–858.

Fisher, Ronald Aylmer. 1958. *The genetical theory of natural selection.*

Fix, Evelyn, & Hodges Jr, Joseph L. 1951. *Discriminatory analysis-nonparametric discrimination: consistency properties.* Tech. rept. DTIC Document.

Fodor, Imola K. 2002. *A survey of dimension reduction techniques.*

Fuhr, Norbert, Hartmann, Stephan, Lustig, Gerhard, Schwantner, Michael, Tzeras, Kostas, & Knorz, Gerhard. 1991. *AIR, X: a rule based multistage indexing system for large subject fields.* Citeseer.

Fujino, Akinori, Ueda, Naonori, & Saito, Kazumi. 2005. A hybrid generative/discriminative approach to semi-supervised classifier design. *Page 764 of: Proceedings of the National Conference on Artificial Intelligence*, vol. 20. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Galavotti, Luigi, Sebastiani, Fabrizio, & Simi, Maria. 2000. Experiments on the use of feature selection and negative evidence in automated text categorization. *Pages 59–68 of: Research and Advanced Technology for Digital Libraries.* Springer.

Ganiz, Murat Can, George, Cibin, & Pottenger, William M. 2011. Higher order Naive Bayes: a novel non-IID approach to text classification. *Knowledge and Data Engineering, IEEE Transactions on*, **23**(7), 1022–1034.

Gasanova, T., Sergienko, R., Minker, W., & Zhukov, E. 2013a. A New Method for Natural Language Call Routing Problem Solving. *Bulletin of Siberian State Aerospace University named after academician M.F. Reshetnev.*

Gasanova, T., Sergienko, R., Minker, W., & Semenkin, E. 2013b. Text Categorization by Coevolutionary Genetic Algorithm. *Bulletin of Siberian State Aerospace University named after academician M.F. Reshetnev.*

Gasanova, T., Sergienko, R., Semenkin, E., & Minker, W. 2014a. Dimension Reduction with Coevolutionary Genetic Algorithm for Text Classification. *In: Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO).*

Gasanova, Tatiana, Zhukov, Eugene, Sergienko, Roman, Semenkin, Eugene, & Minker, Wolfgang. 2013c. A Semi-supervised Approach for Natural Language Call Routing. *Proceedings of the SIGDIAL.*

Gasanova, Tatiana, Sergienko, Roman, Akhmedova, Shakhnaz, Semenkin, Eugene, & Minker, Wolfgang. 2014b. Opinion Mining and Topic Categorization with Novel Term Weighting. *ACL 2014*, 84.

Ghag, Kranti, & Shah, Ketan. 2014. SentiTFIDF–Sentiment Classification using Relative Term Frequency Inverse Document Frequency. *International Journal of Advanced Computer Science & Applications*, **5**(2).

Goldman, Sally, & Zhou, Yan. 2000. Enhancing supervised learning with unlabeled data. *Pages 327–334 of: ICML*. Citeseer.

Gorin, Allen L, Riccardi, Giuseppe, & Wright, Jeremy H. 1997. How may I help you? *Speech communication*, **23**(1), 113–127.

Griffiths, Thomas L, Steyvers, Mark, & Tenenbaum, Joshua B. 2007. Topics in semantic representation. *Psychological review*, **114**(2), 211.

Grzywaczewski, Adam, & Iqbal, Rahat. 2012. Task-specific information retrieval systems for software engineers. *Journal of Computer and System Sciences*, **78**(4), 1204–1218.

Han, Eui-Hong Sam, Karypis, George, & Kumar, Vipin. 2001. *Text categorization using weight adjusted k-nearest neighbor classification*. Springer.

Harris, Zellig S. 1954. Distributional structure. *Word*.

Hassan, Samer, Mihalcea, Rada, & Banea, Carmen. 2007. Random walk term weighting for improved text classification. *International Journal of Semantic Computing*, **1**(04), 421–439.

Hinton, Geoffrey E, & Salakhutdinov, Ruslan R. 2006. Reducing the dimensionality of data with neural networks. *Science*, **313**(5786), 504–507.

Holub, A, Welling, M, & Perona, P. 2005. Exploiting unlabelled data for hybrid object classification. *In: Proc. Neural Information Processing Systems, Workshop Inter-Class Transfer*, vol. 7.

Huang, Fei, & Yates, Alexander. 2009. Distributional representations for handling sparsity in supervised sequence-labeling. *Pages 495–503 of: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics.

Hull, David. 1994. Improving text retrieval for the routing problem using latent semantic indexing. *Pages 282–291 of: SIGIR*. Springer.

Hurault-Plantet, Martine, Berthelin, Jean-Baptiste, El Ayari, Sarra, Grouin, Cyril, Loiseau, Sylvain, & Paroubek, Patrick. 2008. Resultats de l'edition 2008 du DEfi Fouille de Textes. *Actes du quatrieme DEfi Fouille de Textes*, 13.

Iqbal, Rahat, & Shah, Nazaraf. 2012. Information retrieval, decision making process and user needs. *Journal of Computer and System Sciences*, **78**(4), 1158–1159.

Ishibuchi, Hisao, Nakashima, Tomoharu, & Murata, Tadahiko. 1999. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, **29**(5), 601–618.

Joachims, Thorsten. 1998. *Text categorization with support vector machines: Learning with many relevant features*. Springer.

John, George H, Kohavi, Ron, Pfleger, Karl, *et al.* 1994. Irrelevant Features and the Subset Selection Problem. *Pages 121–129 of: ICML*, vol. 94.

Jones, Rosie. 2005. *Learning to extract entities from labeled and unlabeled text*. Ph.D. thesis, University of Utah.

Karabulut, Esra Mahsereci, Özel, Selma Ayşe, & İbrikçi, Turgay. 2012. A comparative study on the effect of feature selection on classification accuracy. *Procedia Technology*, **1**, 323–327.

Karavasilis, Ioannis, Zafiropoulos, Kostas, & Vrana, Vasiliki. 2010. A model for investigating e-governance adoption using tam and doi. *International Journal of Knowledge Society Research (IJKSR)*, **1**(3), 71–86.

Karypis, George, & Kumar, Vipin. 1998. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN*.

Kennedy, James, & Eberhart, Russell C. 1997. A discrete binary version of the particle swarm algorithm. *Pages 4104–4108 of: Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5. IEEE.

Kennedy, James, Eberhart, Russell, *et al.* 1995. Particle swarm optimization. *Pages 1942–1948 of: Proceedings of IEEE international conference on neural networks*, vol. 4. Perth, Australia.

Ko, Youngjoong. 2012. A study of term weighting schemes using class information for text classification. *Pages 1029–1030 of: Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Koo, Terry, Carreras, Xavier, & Collins, Michael. 2008. Simple semi-supervised dependency parsing.

Koza, John R, & James, P. 1992. *Rice, Genetic programming (videotape): the movie.*

Lam, Wai, Low, Kon Fan, & Ho, Chao Yang. 1997. Using a Bayesian network induction approach for text categorization. *Pages 745–750 of: IJCAI (1).*

Landauer, Thomas K, Foltz, Peter W, & Laham, Darrell. 1998. An introduction to latent semantic analysis. *Discourse processes*, **25**(2-3), 259–284.

Lang, Ken. 1995. Newsweeder: Learning to filter netnews. *In: In Proceedings of the Twelfth International Conference on Machine Learning.* Citeseer.

Larkey, Leah S. 1998. Automatic essay grading using text categorization techniques. *Pages 90–95 of: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

Larkey, Leah S, & Croft, W Bruce. 1996. Combining classifiers in text categorization. *Pages 289–297 of: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

Lee, Chin-Hui, Carpenter, Bob, Chou, Wu, Chu-Carroll, Jennifer, Reichl, Wolfgang, Saad, Antoine, & Zhou, Qiru. 2000. On natural language call routing. *Speech Communication*, **31**(4), 309–320.

Lewis, David D. 1992. An evaluation of phrasal and clustered representations on a text categorization task. *Pages 37–50 of: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

Lewis, David D. 1995. Evaluating and optimizing autonomous text classification systems. *Pages 246–254 of: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

Lewis, David D. 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. *Pages 4–15 of: Machine learning: ECML-98.* Springer.

Lewis, David D, & Knowles, Kimberly A. 1997. Threading electronic mail: A preliminary study. *Information processing & management*, **33**(2), 209–217.

Lewis, David D, Schapire, Robert E, Callan, James P, & Papka, Ron. 1996. Training algorithms for linear text classifiers. *Pages 298–306 of: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

Li, Decong, Li, Sujian, Li, Wenjie, Wang, Wei, & Qu, Weiguang. 2010. A semi-supervised key phrase extraction approach: learning from title phrases through a document semantic network. *Pages 296–300 of: Proceedings of the ACL 2010 Conference Short Papers*. Association for Computational Linguistics.

Liang, Jing J, Zhigang, Shang, & Zhihui, Li. 2010. Coevolutionary comprehensive learning particle swarm optimizer. *Pages 1–8 of: Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE.

Liang, Percy. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

Liu, B. 2009. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*.

Liu, Bing, & Zhang, Lei. 2012. A survey of opinion mining and sentiment analysis. *Pages 415–463 of: Mining Text Data*. Springer.

Liu, Dehua, Tu, Bojun, Qian, Hui, & Zhang, Zhihua. 2013. Large-Scale Hierarchical Classification via Stochastic Perceptron. *In: Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Luo, Qiming, Chen, Enhong, & Xiong, Hui. 2011. A semantic term weighting scheme for text categorization. *Expert Systems with Applications*, **38**(10), 12708–12716.

Maeireizo, Beatriz, Litman, Diane, & Hwa, Rebecca. 2004. Co-training for predicting emotions with spoken dialogue data. *Page 28 of: Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics.

Man Lan, Chew Lim Tan, Jian Su, & Lu, Yue. 2009. Supervised and Traditional Term Weighting Methods for Automatic Text Categorization. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*.

Mandelbrot, Benoıt B. 1961. On the theory of word frequencies and on related Markovian models of discourse.

Manning, Christopher D, & Schütze, Hinrich. 1999. *Foundations of statistical natural language processing*. MIT press.

Martín-Valdivia, Maria Teresa, Ureña-López, LA, & García-Vega, M. 2007. The learning vector quantization algorithm applied to automatic text classification tasks. *Neural Networks*, **20**(6), 748–756.

Masand, Brij. 1994. *Optimising confidence of text classification by evolution of symbolic expressions.* MIT Press, Cambridge, MA.

Masand, Brij, Linoff, Gordon, & Waltz, David. 1992. Classifying news stories using memory based reasoning. *Pages 59–65 of: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

McCallum, Andrew Kachites. 1996. *Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering.*

McCallum, Andrew Kachites. 2002. Mallet: A machine learning for language toolkit.

Meena, M Janaki, & Chandran, KR. 2009. Naive Bayes text classification with positive features selected by statistical method. *Pages 28–33 of: Advanced Computing, 2009. ICAC 2009. First International Conference on.* IEEE.

Miao, Yun-Qian, & Kamel, Mohamed. 2011. Pairwise optimized Rocchio algorithm for text categorization. *Pattern Recognition Letters*, **32**(2), 375–382.

Miller, David RH, Leek, Tim, & Schwartz, Richard M. 1999. A hidden Markov model information retrieval system. *Pages 214–221 of: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

Minker, W, Albalate, A, Buhler, D, Pittermann, A, Pittermann, J, Strauss, P-M, & Zaykovskiy, D. 2006. Recent trends in spoken language dialogue systems. *Pages 1–2 of: Information & Communications Technology, 2006. ICICT'06. ITI 4th International Conference on.* IEEE.

Mitchell, Tom. 1999. The role of unlabeled data in supervised learning. *Pages 2–11 of: Proceedings of the sixth international colloquium on cognitive science.* Citeseer.

Mnih, Andriy, & Hinton, Geoffrey. 2007. Three new graphical models for statistical language modelling. *Pages 641–648 of: Proceedings of the 24th international conference on Machine learning.* ACM.

Momtazi, Saeedeh, & Klakow, Dietrich. 2009. A word clustering approach for language model-based sentence retrieval in question answering systems. *Pages 1911–1914 of: Proceedings of the 18th ACM conference on Information and knowledge management.* ACM.

Moulinier, Isabelle, & Ganascia, Jean-Gabriel. 1996. Applying an existing machine learning algorithm to text categorization. *Pages 343–354 of: Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing.* Springer.

Neto, Ajalmar RR, & Barreto, Guilherme A. 2013. Opposite Maps: Vector Quantization Algorithms for Building Reduced-Set SVM and LSSVM Classifiers. *Neural processing letters*, **37**(1), 3–19.

Ng, Hwee Tou, Goh, Wei Boon, & Low, Kok Leong. Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization.

Ng, Hwee Tou, Goh, Wei Boon, & Low, Kok Leong. 1997. Feature selection, perceptron learning, and a usability case study for text categorization. *Pages 67–73 of: ACM SIGIR Forum*, vol. 31. ACM.

Nigam, Kamal, & Ghani, Rayid. 2000. Analyzing the effectiveness and applicability of co-training. *Pages 86–93 of: Proceedings of the ninth international conference on Information and knowledge management*. ACM.

Nigam, Kamal, McCallum, Andrew Kachites, Thrun, Sebastian, & Mitchell, Tom. 2000. Text classification from labeled and unlabeled documents using EM. *Machine learning*, **39**(2-3), 103–134.

Ong, Thian-Huat, & Chen, Hsinchun. 1999. Updateable PAT-Tree approach to Chinese key phrase extraction using mutual information: A linguistic foundation for knowledge management.

Palus, Sebastian, Brodka, Piotr, & Kazienko, Przemyslaw. 2011. Evaluation of organization structure based on email interactions. *International Journal of Knowledge Society Research (IJKSR)*, **2**(1), 1–13.

Park, Sun, Sharma, Ronesh Asnil, Park, Jin Gwan, Jeong, Min A, Shin, Jun Woo, & Lee, Seong Ro. 2013. Document Clustering using Reweighted Term.

Porter, Martin F. 1980. An algorithm for suffix stripping. *Program: electronic library and information systems*, **14**(3), 130–137.

Potter, Mitchell A, & De Jong, Kenneth A. 2000. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, **8**(1), 1–29.

Quan, Xiaojun, Wenyin, Liu, & Qiu, Bite. 2011. Term weighting schemes for question categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **33**(5), 1009–1021.

Ratinov, Lev, & Roth, Dan. 2009. Design challenges and misconceptions in named entity recognition. *Pages 147–155 of: Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

Reed, Joel W, Jiao, Yu, Potok, Thomas E, Klump, Brian A, Elmore, Mark T, & Hurson, Ali R. 2006. TF-ICF: A new term weighting scheme for clustering dynamic data streams. *Pages 258–263 of: Machine Learning and Applications, 2006. ICMLA'06. 5th International Conference on.* IEEE.

Riloff, Ellen. 1995. Little words can make a big difference for text classification. *Pages 130–136 of: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM.

Riloff, Ellen, Wiebe, Janyce, & Wilson, Theresa. 2003. Learning subjective nouns using extraction pattern bootstrapping. *Pages 25–32 of: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4.* Association for Computational Linguistics.

Robertson, Stephen. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation*, **60**(5), 503–520.

Robertson, Stephen E, & Jones, K Sparck. 1976. Relevance weighting of search terms. *Journal of the American Society for Information science*, **27**(3), 129–146.

Robertson, Stephen E, Walker, Steve, Jones, Susan, Hancock-Beaulieu, Micheline M, Gatford, Mike, *et al.* 1995. Okapi at TREC-3. *NIST SPECIAL PUBLICATION SP*, 109–109.

Rocchio, Joseph John. 1971. Relevance feedback in information retrieval.

Roman Sergienko, Tatiana Gasanova, Eugene Semenkin, & Minker, Wolfgang. 2016. Collectives of Term Weighting Methods for Natural Language Call Routing. *Lecture Notes in Electrical Engineering*, **Informatics in Control Automation and Robotics - ICINCO 2014**.

Ropero, Jorge, Gómez, Ariel, León, Carlos, & Carrasco, Alejandro. 2009. Term Weighting: Novel Fuzzy Logic based Method Vs. Classical TF-IDF Method for Web Information Extraction. *Pages 130–137 of: ICEIS (2).*

Rosenberg, Chuck, Hebert, Martial, & Schneiderman, Henry. 2005. Semi-supervised self-training of object detection models.

Rosenblatt, Frank. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, **65**(6), 386.

Rousseau, François, & Vazirgiannis, Michalis. 2013. Graph-of-word and TW-IDF: new approach to ad hoc IR. *Pages 59–68 of: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management.* ACM.

Ruiz, Miguel E, & Srinivasan, Padmini. 1999. Hierarchical neural networks for text categorization (poster abstract). *Pages 281–282 of: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Rushdi Saleh, Mohammed, Martín-Valdivia, Maria Teresa, Montejo-Ráez, A, & Ureña-López, LA. 2011. Experiments with SVM to classify opinions in different domains. *Expert Systems with Applications*, **38**(12), 14799–14804.

Sable, Carl L, & Hatzivassiloglou, Vasileios. 2000. Text-based approaches for non-topical image categorization. *International Journal on Digital Libraries*, **3**(3), 261–275.

Sahami, Mehran, Dumais, Susan, Heckerman, David, & Horvitz, Eric. 1998. A Bayesian approach to filtering junk e-mail. *Pages 98–105 of: Learning for Text Categorization: Papers from the 1998 workshop*, vol. 62.

Salton, G., & Buckley, C. 1988. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*.

Sasaki, Takahiro, & Tokoro, Mario. 1999. Evolving learnable neural networks under changing environments with various rates of inheritance of acquired characters: comparison of Darwinian and Lamarckian evolution. *Artificial Life*, **5**(3), 203–223.

Schapire, Robert E, & Singer, Yoram. 2000. BoosTexter: A boosting-based system for text categorization. *Machine learning*, **39**(2-3), 135–168.

Schmid, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. *Pages 44–49 of: Proceedings of international conference on new methods in language processing*, vol. 12. Manchester, UK.

Schwenk, Holger, & Gauvain, Jean-Luc. 2002. Connectionist language modeling for large vocabulary continuous speech recognition. *Pages I–765 of: Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 1. IEEE.

Sebastiani, Fabrizio. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, **34**(1), 1–47.

Seeger, Matthias, *et al.* 2001. *Learning with labeled and unlabeled data*. Tech. rept. technical report, University of Edinburgh.

Sergienko, R., Minker, W., Semenkin, E., & Gasanova, T. 2013. Call Routing Problem Solving Method Based on a New Term Relevance Estimation. *Program Products and Systems*.

Sergienko, R., Gasanova, T., Semenkin, E., & Minker, W. 2014. Text Categorization Methods Application for Natural Language Call Routing. *Special Session on Adaptive Stochastic Algorithms and their Applications at Human-Machine Interfaces - ASAAHMI 2014 Within the 11th International Conference on Informatics in Control, Automation and Robotics - ICINCO 2014.*

Shafait, Faisal, Reif, Matthias, Kofler, Christian, & Breuel, Thomas M. 2010. Pattern Recognition Engineering. *In: RapidMiner Community Meeting and Conference*, vol. 9.

Slonim, Noam, & Tishby, Naftali. 2001. The power of word clusters for text classification. *In: 23rd European Colloquium on Information Retrieval Research*, vol. 1.

Smeshko, Y., & Gasanova, T. 2012. Evolutionary Modification of Fuzzy C-means Algorithm with Automatic Selection of the Number of Clusters for Speech Utterance Categorization. *Bulletin of Siberian State Aerospace University named after academician M.F. Reshetnev Reshetnev.*

Song, Young-In, Han, Kyoung-Soo, & Rim, Hae-Chang. 2004. A term weighting method based on lexical chain for automatic summarization. *Pages 636–639 of: Computational Linguistics and Intelligent Text Processing.* Springer.

Soucy, P., & Mineau, G.W. 2005. Beyond TFIDF Weighting for Text Categorization in the Vector Space Model. *In: Proc. Int'l Joint Conf. Artificial Intelligence,.*

Suendermann, David, Liscombe, Jackson, Dayanidhi, Krishna, & Pieraccini, Roberto. 2009. A handsome set of metrics to measure utterance classification performance in spoken dialog systems. *Pages 349–356 of: Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue.* Association for Computational Linguistics.

Suzuki, Jun, & Isozaki, Hideki. 2008. Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. *Pages 665–673 of: ACL.* Citeseer.

Suzuki, Jun, Isozaki, Hideki, Carreras, Xavier, & Collins, Michael. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. *Pages 551–560 of: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2.* Association for Computational Linguistics.

Trappey, Amy JC, Hsu, Fu-Chiang, Trappey, Charles V, & Lin, Chia-I. 2006. Development of a patent document classification and search platform using a back-propagation network. *Expert Systems with Applications*, **31**(4), 755–765.

Turian, Joseph, Ratinov, Lev, & Bengio, Yoshua. 2010. Word representations: a simple and general method for semi-supervised learning. *Pages 384–394 of: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Turney, Peter D. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, **2**(4), 303–336.

Tzeras, Kostas, & Hartmann, Stephan. 1993. Automatic indexing based on Bayesian inference networks. *Pages 22–35 of: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Umer, Muhammad Fahad, & Khiyal, M. 2007. Classification of Textual Documents using Learning Vector Quantization. *Information Technology Journal*, **6**(1).

Vapnik, Vladimir N, & Chervonenkis, A Ja. 1974. Theory of pattern recognition.

Vieira, Susana M, Sousa, JoãO, & Kaymak, Uzay. 2012. Fuzzy criteria for feature selection. *Fuzzy Sets and Systems*, **189**(1), 1–18.

Wallace, David L. 1983. Comment. *Journal of the American Statistical Association*, **78**(383), 569–576.

Ward Jr, Joe H. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, **58**(301), 236–244.

Weinberger, Kilian Q, & Saul, Lawrence K. 2009. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, **10**, 207–244.

Wiener, Erik, Pedersen, Jan O, Weigend, Andreas S, *et al.* 1995. A neural network approach to topic spotting. *Pages 317–332 of: Proceedings of SDAIR-95, 4th annual symposium on document analysis and information retrieval*. Citeseer.

Wilson, Edwin B. 1927. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, **22**(158), 209–212.

Wright, Jeremy H, Gorin, Allen L, & Riccardi, Giuseppe. 1997. Automatic acquisition of salient grammar fragments for call-type classification. *In: EUROSPEECH*.

Yang, Chenguang, Tu, Xuyan, & Chen, Jie. 2007. Algorithm of marriage in honey bees optimization based on the wolf pack search. *Pages 462–467 of: Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on*. IEEE.

Yang, Hui, & Chua, Tat-Seng. 2004a. Effectiveness of web page classification on finding list answers. *Pages 522–523 of: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Yang, Hui, & Chua, Tat-Seng. 2004b. Web-based list question answering. *Page 1277 of: Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics.

Yang, Xin-She. 2009. Firefly algorithms for multimodal optimization. *Pages 169–178 of: Stochastic algorithms: foundations and applications*. Springer.

Yang, Xin-She. 2010. A new metaheuristic bat-inspired algorithm. *Pages 65–74 of: Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer.

Yang, Xin-She, & Deb, Suash. 2009. Cuckoo search via Lévy flights. *Pages 210–214 of: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. IEEE.

Yang, Yiming, & Liu, Xin. 1999. A re-examination of text categorization methods. *Pages 42–49 of: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Yang, Yiming, & Pedersen, Jan O. 1997. A comparative study on feature selection in text categorization. *Pages 412–420 of: ICML*, vol. 97.

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. *Pages 189–196 of: Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Yu, Bo, Xu, Zong-ben, & Li, Cheng-hua. 2008. Latent semantic analysis for text categorization using neural network. *Knowledge-Based Systems*, **21**(8), 900–904.

Z.-H. Deng, S.-W. Tang, D.-Q. Yang M. Zhang L.-Y. Li, & Xie, K.Q. 2004. A Comparative Study on Feature Weight in Text Categorization. *In: Proc. Asia-Pacific Web Conf.*

Zhang, Jin, & Nguyen, Tien N. 2005. A new term significance weighting approach. *Journal of Intelligent Information Systems*, **24**(1), 61–85.

Zhang, Min-Ling, & Zhou, Zhi-Hua. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, **18**(10), 1338–1351.

Zhou, Zhi-Hua, Zhan, De-Chuan, & Yang, Qiang. 2007. Semi-supervised learning with very few labeled training examples. *Page 675 of: Proceedings of the national conference on*

artificial intelligence, vol. 22. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Zhu, Xiaojin. 2006. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, **2**, 3.