TEXT CLASSIFICATON IN TURKISH MARKETING DOMAIN AND CONTEXT-SENSITIVE AD DISTRIBUTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MELİH ENGİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

FEBRUARY 2009

Approval of the thesis:

TEXT CLASSIFICATON IN TURKISH MARKETING DOMAIN AND CONTEXT-SENSITIVE AD DISTRIBUTION

submitted by **MELİH ENGİN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by;

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Müslim Bozyiğit
Head of Department, **Computer Engineering** _____

Assist. Prof. Dr. Tolga Can
Superviser, **Computer Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Volkan Atalay
Computer Engineering Dept., METU _____

Assist. Prof. Dr. Tolga Can
Computer Engineering Dept., METU _____

Assist. Prof. Dr. Pinar Senkul
Computer Engineering Dept., METU _____

Dr. Onur Tolga Şehitoğlu
Computer Engineering Dept., METU _____

Dr. Güven Fidan
General Manager, AGMLAB _____

Date: 12.02.2009

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Surname: Melih Engin

Signature :

ABSTRACT

TEXT CLASSIFICATON IN TURKISH MARKETING DOMAIN AND CONTEXT-SENSITIVE AD DISTRIBUTION

Engin, Melih

M.S. Department of Computer Engineering

Supervisor: Assist. Prof. Dr. Tolga Can

February 2009, 53 pages

Online advertising has a continuously increasing popularity. Target audience of this new advertising method is huge. Additionally, there is another rapidly growing and crowded group related to internet advertising that consists of web publishers. Contextual advertising systems make it easier for publishers to present online ads on their web sites, since these online marketing systems automatically divert ads to web sites with related contents. Web publishers join ad networks and gain revenue by enabling ads to be displayed on their sites. Therefore, the accuracy of automated ad systems in determining ad-context relevance is crucial.

In this thesis we construct a method for semantic classification of web site contexts in Turkish language and develop an ad serving system to display context related ads on web documents. The classification method uses both semantic and statistical techniques. The method is supervised, and therefore, needs processed sample data for learning classification rules. Therefore, we generate a Turkish marketing dataset and use it in our classification approaches. We form successful classification methods using different feature spaces and support vector machine configurations. Our results present a good comparison between these methods.

Keywords: Text Classification, Data Mining, Machine Learning, Artificial Intelligence, Information Retrieval

ÖZ

TÜRKÇE PAZARLAMA ALANINDA TEKST SINIFLANDIRMASI VE İÇERİK DUYARLI REKLAM DAĞITIMI

Engin, Melih
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Yard. Doç. Dr. Tolga Can

Şubat 2009, 53 sayfa

Günümüzde internet reklamcılığı hızla artan bir popülerliğe sahiptir. İnternet reklamları büyük kitlelere ulaşabiliyor. Reklamların hedef kitleleri yanısıra internet reklamcılığı ile yakından ilgili diğer bir grup da internet yayıncılarıdır. Otomatik internet reklamcılığı sistemlerinin gelişmesi ve bu sistemlerin içeriğe göre reklamları sitelere dağıtabilmeleri ile yayıncılar kolayca sitelerinde reklamlar sunabilir hale geldiler. İnternet yayıncıları reklam ağlarına katılıp reklamların kendi sitelerinde de yayınlanmasını sağlayarak gelir elde edebiliyor. Bu noktada otomatik çalışan reklam sistemlerinin içerik belirlemede sahip oldukları doğruluk yüzdesi çok önemli bir hal almıştır.

Bu tez çalışmasında Türkçe İnternet sitesi içeriklerini anlamsal sınıflandırmak için bir method ve İnternet sayfalarına içerikle ilgili reklam dağıtmak için bir sistem oluşturuyoruz. Sınıflandırma methodu anlamsal ve istatistiksel teknikler kullanmaktadır. Öğrenmeye dayalı bir sınıflandırma yöntemi önermekteyiz. Sınıflandırma kurallarının öğrenilmesi için işlenmiş bir örnek veri kümesi gerekli. Bu sebeple Türkçe bir pazarlama verisi oluşturduk ve bunu sınıflandırma yöntemlerimizde kullandık. Farklı özellik uzayları ve destek vektör makinesi ayarları kullanarak başarılı sınıflandırma metotları oluşturduk. Aldığımız sonuçlar oluşturduğumuz bu metotlar arasında iyi bir karşılaştırma olanağı sağladı.

Anahtar Kelimeler: Metin Sınıflandırması, Veri Madenciliği, Makine Öğrenimi, Yapay Zeka, Bilgi Erişimi

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Massive online information, rapidly growing number of web documents on the Internet has made data organization techniques very important in the past decade. There is a need for elegant and specialized classification techniques for many domains in order to make information more manageable and reachable. General approaches suffer in answering localized needs. This statement is highly valid for learning methods like classification.

One important example of the need for classification of web documents is online advertising. Limited budgets require using the most cost efficient systems on advertising. Parameters like click through rate, conversion count have primary importance in evaluating effectiveness [18, 19]. Systems taking these performance parameters like pay per click (PPC) and pay per action (PPA) into account have gained popularity in the past few years. The main reason is that these systems provide their publishers the option of paying only for the portion of their advertising that meets its objectives. These objectives are defined as an Internet user clicking on a banner or completing a predefined action on the site that has been redirected to after clicking on the banner. However these systems need to find related users for their banners in order to increase the number of completed objectives. In finding real target audience on the Internet, additional features like context sensitivity or behavioral targeting are crucial.

Context sensitive systems highly dominate online advertising in today's world [20]. They deliver related advertisements according to content. In order to achieve this, they periodically crawl online documents and classify them.

1.1 Problem Definition

Having stated the fact that general approaches suffer in answering localized needs, our problem can be defined as developing a specialized approach for classifying Turkish web documents in the marketing domain. We need to develop classification techniques specialized for our target language and target domain to obtain a high performance and

accurate classifier. Our data consist of Turkish web documents in the marketing domain. We should use characteristics of web documents such as weighting HTML tags like *title*, *meta* and *keyword* tags. Since, we concentrate on the Turkish language, stemmers and sentence rules for Turkish can be used. Additionally, marketing domain specific characteristics can be highlighted by selecting data sources representing the domain best.

Our problem can be stated formally as follows: Given an input Turkish web document *d* in the marketing domain and *k* merchandise classes, classify *d* as one of the merchandise categories. This is a multi-class classification problem and in this thesis, we propose to use a supervised learning framework. We also propose a variety of feature extraction techniques which are specific to the studied language and domain. Note that feature extraction is crucial to the success of any supervised learning technique.

1.2 Related Work

Classification of web documents can highly benefit from text classification, feature extraction and web mining approaches. There have been many studies on these fields in the past decade [4, 5, 21].

Text classification samples are mainly documents containing a large number of words and sentences. These documents need to be transformed into a feature set representation suitable for the learning algorithm of the classification method. Information retrieval researches suggest that word stems (denoted as *root features* in the rest of the thesis) work well as representation units and their ordering in a text is of minor importance for many tasks [4]. Researches also provide both theoretical and empirical evidence that Support Vector Machines (SVMs) are very well suited for text categorization [4]. Theoretical analyses conclude that SVMs acknowledge the particular properties of text: high dimensional feature spaces, few irrelevant features in a dense concept vector, and sparse instance vectors [4].

A large feature set also limits the optimization capabilities due to performance limitations no matter which classification method is used. Yıldız et al. worked on a new feature extraction method on Turkish text documents to decrease the number of features to the number of classes [1]. This study has shown that transforming root features into class features through a weighting process does not decrease the accuracy of the classification approach. In this thesis, we wanted to benefit from the results of this study

using a more general, larger dataset, with a more detailed weighting process. We developed a successful classification method for comparison with other classification methodologies we study in this thesis.

On the other hand there are studies that prove singular value decomposition (SVD) is a successful method in reducing the number of features in text classification [6]. SVD is a well-known literature method used for feature reduction [22]. SVD is a good candidate for comparison with the method we mention in the previous paragraph.

Weighting process of word stems within the documents is a crucial step. Weighting extracts semantic information on documents using statistical methods when semantic methods are not affordable. Studies on information retrieval, extraction fields show that traditional classification approaches augmented with linguistic information at a suitable level increase success rates [3, 9]. There are also text categorization studies showing that distribution of word stems within a document determine the level of its semantic weight within the document [2]. These studies show that embedding statistical methods with semantic ones at an affordable level is a must. A more complete augmentation is possible using a semantic lexicon. WordNet [7] is a well known semantic lexicon for the English language. A Turkish WordNet project within the BalkaNet project also exists [8].

These studies state useful facts on text classification, information retrieval, information extraction, machine learning, natural language processing, web mining and other related fields. These results need to be organized with some additional work to form building structures for a successful classification method. We aimed to combine the important features of these works and to extend them by taking into account special features of our specific problem domain.

1.3 Contributions and Outline of the Thesis

Our contribution in this these can be summarized as follows:

1. Existing feature extraction methods such as TF-IDF and distributional features are enhanced with web document structure, sentence characteristics of Turkish and additional weighting formulations.

2. Existing category feature extraction methods are re-implemented and enhanced with a new category TF-IDF definition and category weighting formulas.
3. Turkish marketing domain datasets in different feature spaces are built that can be used in future work.
4. Performances of several SVM configurations are tested on different datasets in terms of accuracy and speed.

The rest of the thesis is organized as follows. Chapter 2 gives theoretical background on classification methods and the tools, packages we have used in the implementation. Chapter 3 states issues we have considered in determining our data sources, designing our applications and determining our limitations due to performance. In Chapter 4, we list the main phases of our classification framework and explain details of algorithms and implementations for each phase. In Chapter 5 we explain the tests we have run using the applications we have built in the implementation part. Chapter 6 concludes with an overview of our study and discussion of our results.

CHAPTER 2

CLASSIFICATION BACKGROUND

Our study mainly involves a feature extraction phase, training of a supervised classifier phase and cooperative algorithms to optimize the feature space. Feature extraction is a determinative phase on the success of a classification method. Text classification procedures use words to form their features. Term frequency (TF), inverse document frequency (IDF) parameters help in avoiding semantically ineffective words like stop words. TF-IDF are explained in detail in the Methodologies subsection of this chapter. The classification algorithm we used is Support Vector Machine (SVM) [4, 10]. SVMs are also presented with their kernel functions and other key parameters. We also used a feature space transformation technique called Principal Component Analysis (PCA) [22] for comparison reasons. PCA and the factorization method of Singular Value Decomposition (SVD) [22] underlying it are also described.

We also used various available applications in different phases of our implementation. We used ZEMBEREK [16] tool in the feature extraction phase; LIBSVM [12], LIBLINEAR [13] tools on the classification phase and investigated WEKA, JAMA packages for feature space transformation tasks. These tools and packages are also presented in the Tools subsection.

2.1 Methodologies

In this section, we describe the main methodologies we used in our thesis.

2.1.1 *Support Vector Machines (SVMs)*

SVM is a powerful supervised technique for data classification. Structural Risk Minimization principle underlies the SVM approach [4]. This principle aims to find a hypothesis that minimizes the true error [4]. Another important property of SVM is preprocessing the data to represent patterns in a higher dimension. With an appropriate

nonlinear mapping function and a sufficiently high dimension, data from two categories can always be separated by a hyperplane [11]. Training a SVM consists of finding the optimal hyperplane, that is, the one with the maximum distance from the nearest training patterns [11].

Given a training set of instance-label pairs (x_i, y_i), i=1,...,l where $x_i \in \mathrm{R_n}$ and $y \in \{1,-1\}^l$, the support vector machines require the solution of the following optimization problem;

$$1/2\mathrm{w}^T w + C\sum_{i=1}^{l}\xi_i$$

$$\text{subject to } y_i\left(w^T\Phi(x_i)+b\right)\geq 1-\xi_i \qquad (1)$$

$$\xi \geq 0,$$

Here training vectors x_i's are mapped into a higher dimensional space by the function. Then SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. C > 0 is the penalty parameter of the error term. Furthermore, $K(x_i, x_j) \equiv \Phi(x_i)^T\Phi(x_j)$ is called the kernel function [10]. Four most widely used kernels are linear, polynomial, radial basis function and sigmoid kernels [12]. Note that in this definition SVM solves a binary classification problem. However, there are a number of techniques to use a number of binary SVM classifiers to solve a multi-class classification problem [12].

SVMs can solve various classification problems successfully. The training algorithm can learn a separating hyperplane independent of the dimensionality of the feature space. The algorithm measure the complexity of hypotheses based on the margin with which it separates the data without using the number of features [4]. In this way the algorithm covers the case of large number of features, if the data is separable with a margin using the kernel functions from the hypothesis space.

We made practical use of two kernel functions; linear and radial basis function. Radial basis function (RBF) kernel is a successful standard kernel function. The RBF kernel can transform instances to a higher dimensional space nonlinearly. This enables the classifier to learn nonlinear relations between class labels and the features. It has practical advantages on polynomial and sigmoid kernels. However, linear kernel is more preferable than RBF kernel in case of large number of features [10]. Below we give some standard kernel functions used in the literature.

Linear: $K(x_i, x_j) = x_i^T x_j$ (2)

Polynomial: $K(x_i, x_j) = (x_i^T x_j + r)^d, \quad \gamma > 0$ (3)

RBF: $K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \quad \gamma > 0$ (4)

Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$ (5)

Here γ, r and d are kernel parameters [10].

2.1.2 *Singular Value Decomposition*

SVD is a powerful technique used in matrix analyses [22]. Using SVD a matrix is decomposed into a series of transformations like rotation, scaling. It can be used as a data compression technique for rectangular matrices. The theorem is stated as;

$$A = U S V^T \quad (6)$$

Here $\mathbf{A} \in \mathbf{R}_{mxn}$, $\mathbf{m} > \mathbf{n}$, $\mathbf{U} \in \mathbf{R}_{mxn}$, $\mathbf{V} \in \mathbf{R}_{nxm}$ and **S** is a diagonal matrix of size $\mathbf{R}_{nxn}$. Both **U** and **V** are orthogonal. This factorization is called the SVD of **A**. **U** contains a set of orthonormal basis vector directions for **A**, **S** contains the singular values and **V** contains the orthonormal analysis basis vector directions for **A**.

SVD is also used in various applications like solving least squares problems, systems of linear equations, noisy signal filtering and time series analysis [23].

2.1.3 *Principal Component Analysis*

PCA is an unsupervised method to find the most efficient feature space for a dataset [22]. It is a specialized version of SVD.

PCA transformation consists of some main steps. Suppose we have a dataset $\mathbf{D} \in \mathbf{R}^{mxn}$, firstly the dataset should be standardized. We compute the n dimensional mean vector μ containing column means of **D** and σ containing column standard

deviations of **D**. Then the elements of the dataset are mean adjusted using μ and reduced using σ. Let $a_{ij} \in \mathbf{D}$;

$$a_{ij} = \left(a_{ij} - \mu_j\right) / \sigma_j \qquad (7)$$

On the next step, *nxn* dimensional covariance matrix $\mathbf{\Sigma}$ is computed for standardized version of **D**. The eigenvalues and the eigenvectors of $\mathbf{\Sigma}$ are evaluated and sorted according to the eigenvalues. Sorted eigenvalues usually have a significant gap that can be used to eliminate most of them. Finally, corresponding eigenvectors of the selected eigenvalues are used as the new feature set. The process transforms initial dataset into a new feature space with fewer dimensions. PCA provides many classification applications a critical data compression.

2.1.4 *Term Frequency and Inverse Document Frequency*

TF and IDF are weighting parameters in text mining and information retrieval that are usually used together [24]. In text classification our dataset mainly consists of text documents containing collections of word stems. These two parameters mainly determine how important a word is in a document and in the whole corpus.

TF is simply frequency of a word w_i in document d_j. Let D be our dataset and d_j be a set containing l words;

$$d_j \in \mathrm{D},\ d_j = \{w_1, w_2, \ldots, w_l\}$$

$$tf_{ij} = \frac{count\left(w_i, d_j\right)}{\sum_{k=1}^{l} count\left(w_k, d_j\right)} \qquad (8)$$

Here $count(w_i, d_j)$ is the number of occurrences of w_i in d_j.

IDF presents how determinative a word is among the whole corpus. If the percentage of the documents containing the term is high this means the word does not occupy a significant semantic effect on the instances it occur. IDF_i of w_i is given by;

$$idf_i = \log \frac{|D|}{|\{d_j : w_i \in d_j, d_j \in D\}|} \quad (9)$$

Generally TF, IDF is used as a single variable TFIDF in weighting processes. TFIDF of a word w_i in document d_j is simply;

$$tfidf_{ij} = tf_{ij}\, idf_i \quad (10)$$

2.2 Tools

Throughout different phases of our study we used and tested several tools. In this section we will give brief information about these tools.

2.2.1 *LIBSVM*

LIBSVM is an integrated software for support vector classification (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM) [12]. It also supports multi-class classification. It is freely available for use provided that cited properly. It has C++ and Java sources and has interfaces for Python, R, MATLAB, Perl, Ruby, WEKA, Common LISP and LabVIEW. It is developed by Machine Learning Group at National Taiwan University.

LIBSVM provides its users tools for preprocessing their dataset, training their classifier, predicting test set labels and finding optimized parameters for their classifier. LIBSVM training function has various options like setting different kernel functions, different kernel function parameters, and termination criteria. It enables use of linear, polynomial, radial basis function and sigmoid kernels. The tool also enables n-fold cross validation. There are also functions for scaling the dataset, sampling the dataset. The package also provides a grid function which searches for optimal training parameters for the given dataset.

2.2.2 *LIBLINEAR*

LIBLINEAR is a linear classifier for data with millions of instances and features [13]. It

supports L2-regularized logistic regression (LR), L2-loss linear SVM, and L1-loss linear SVM [13]. LIBLINEAR has MATLAB/Octave and Java interfaces. LIBLINEAR is developed by the same group as LIBSVM.

This package has similar usage with LIBSVM. LIBLINEAR gives better performance than LIBSVM at certain situations;

- number of instances in the data set is much less than the number of features
- both number of instances and number of features are very large
- number of instances is very large and much larger than the number of features

At these cases LIBLINEAR is stated to have better performance [12]. We mainly used LIBLINEAR and LIBSVM in the classification phase of our study.

2.2.3 *WEKA Package*

Weka is a collection of machine learning algorithms for data mining tasks [14]. Weka contains tools for data preprocessing, classification, regression, clustering, association rules, and visualization. Weka provides a detailed user interface to facilitate use of the various methods in it. Users can form tests from a series of different algorithms supplied in Weka. These tests can be saved and used later on the study, processed data can be visualized and the users can easily learn valuable information about the approaches they use by the help of this useful interface. One drawback of this multi functional user interface is that it has high performance requirements when working with a large dataset. All functionalities in the interface can also be run using command line Java programs. Weka can also be used as a Java package and can easily be embedded in Java projects.

Weka contains a wide range of classification and data transformation algorithms. We investigated Weka functionality in the feature space transformation and classification phases of our study. However, the size of our dataset caused severe performance problems even if with excessive sampling and external feature elimination.

2.2.4 *JAMA*

JAMA is a basic linear algebra package for Java [15]. JAMA provides various

algorithms on matrix processing and analysis. In JAMA there are implementations for Cholesky decomposition, LU decomposition, QR decomposition, singular value decomposition and eigenvalue decomposition and many other matrix operations. JAMA is a lower level package than Weka for matrix operations. It is more flexible in embedding into our code and can give better performance by means of memory and CPU usage. We mainly used JAMA in our PCA transformation implementation.

2.2.5 *ZEMBEREK*

Zemberek is an open source, platform independent, general purpose Natural Language Processing (NLP) library and tool set designed for Turkic languages, especially Turkish [16]. The project is developed with Java programming language and can be easily embedded into Java projects using the jar files delivered with the package. Two .jar files are needed for using Zemberek for Turkish language. One of them is zemberek-cekirdek.jar which contains the main functions of the Zemberek library and the other is zemberek-tr.jar which contains Turkish specific information and classes. With an additional language .jar file, functionality for the Azerbaijan Turkish or another Turkic language can be added.

CHAPTER 3

DESIGN ISSUES

Before proceeding with the implementation phase of our study, we considered some issues in order to make a good design and stay focused on our problem and domain.

3.1 Domain Specifications

As presented in the previous sections, we want to classify web documents in Turkish in the marketing domain. Therefore, we should analyze marketing domain in depth in order to express this domain entirely in our dataset and utilize more characteristics in our training algorithm. We should define the target documents that our classifier will classify and accordingly we should decide the specifications of our corpus that will train our algorithm best for this objective.

Online marketing domain encloses e-commerce sites, publisher networks, affiliate systems and lead sites. These networks and site groups almost comprise the entire content on the Internet except some governmental and educational sites.

E-commerce sites form a significant group in the online marketing domain. They enable selling and buying of goods and services from almost all categories. They participate in publisher networks and affiliate systems as the largest group of advertisers. These factors provide e-commerce sites a dominant impact on our domain. Further analysis on this site category can give significant information about the whole domain.

E-commerce sites can be used as two sources of information. First one is that they can provide us the whole category tree of products and services that are bought and sold on the Internet. These sites organize items they provide in a categorization. Category trees from several e-commerce sites can form a highly complete tree of all online items with a little manual matching. Secondly these sites form a great source of content related to products. Product descriptions, meta tags, keywords, user comments about products are very valuable for the training phase of a supervised classification technique.

Another important group is Publisher networks. These systems consist of publishers and advertisers. Advertisers in these systems are mainly e-commerce sites as we mentioned above. They create banners related to their products or categories and hire advertisement zones on publisher web sites for a given time frame. Publishers of these networks can be any online content provider. They define advertisement zones on their web sites and sell these zones on their own or in a publisher network.

Affiliate systems are an emerging group in online marketing. They become more strategic as they highlight the importance of efficiency in marketing budgets for both publishers and advertisers. Advertisers of affiliate systems are e-commerce sites and lead sites. Lead sites aim a predefined action by the Internet user directed to their site from one of their banners. This predefined action is called a lead. In fact E-commerce sites are a special case in which the lead is defined as an online payment transaction. Publishers of these networks can be every online content provider just like publisher networks.

In affiliate systems the process is as follows. Advertisers agree to share a certain percentage of the profit they gain from the system with the system and the publisher that provide a conversion to them. Then they load several banners to the system for publishers to chose. Publishers chose banners to publish in their sites according to the past statistics of the banners. Good publishers chose the banners with best earning potential. To sum up this kind of systems provide an ideal “survival of the fittest” environment for both their advertisers and publishers.

In both publisher networks and affiliate systems publishers can be any kind of web site. This highly widens the borders of the marketing domain. A classifier for the marketing domain should aim to classify web documents of any content.

3.2 Performance Issues

We have two objectives in this study. First one is forming a classification procedure in the Turkish marketing domain. The other one is developing an ad distribution system.

Forming a classification procedure requires running several tests on a given dataset. Different classification algorithms should be tested with different parameters. A variety of preprocessing steps should be analyzed. Additionally, the classification step generally requires sophisticated algorithms. Ready implementations, tools that require different data formats as input are usually deployed. These factors require the design

should be modular and flexible. Processing steps should be replaceable with one another. Furthermore, the data should be kept in a flexible, easily accessible system.

There are also considerations brought by developing an ad distribution system. A living context sensitive ad distribution system needs to crawl a certain subset of the web approximately once in a month. According to the content that needs to be crawled, this process should be continuously optimized. Therefore the crawler has to be well separated from the whole system. Additionally, as the crawling size increases, we need a large storage capacity.

Considering these issues we designed the system to be highly modular. There are different processes for crawling, keyword extraction, keyword weighting and feature extraction operations. These processes can be easily optimized and replaced with a better version. For flexibility and accessibility of the raw dataset we stored preprocessed data in a relational database. By this way we could organize and export our data in several formats. We optimized our phases to be run on our deployment machine, which has an AMD x86 64bit processor, 3GBs of memory and a 250GB harddisk. Our feature extraction and classification training processes used nearly all processing power and memory. Our whole study occupied nearly 80GBs of storage. Most of this data consist of the raw dataset.

CHAPTER 4

IMPLEMENTATION

We implemented our own processes in the data collection, feature extraction, data transformation, classification and ad distribution system phases of our study. There are several processes in each phase. We used Java programming language in our implementation. Oracle JDeveloper is used as the development environment. We used MySql 5 server for data storage. There are also *bash* scripts connecting our Java processes and existing tools we used.

A brief review of our phases is as follows;

- The data collection and preprocessing phase consists of crawling web pages, labeling crawled web documents and listing word stems in a web page.
- The feature extraction phase contains processes for evaluating TF-IDF and compactness of word stems in web pages, evaluating TF-IDF of word stems in predefined categories and finding weights of stems in each document and category. In addition, there are processes for exporting processed data as featured instances.
- The classification phase contains *bash* scripts for running several tests using different tools.
- Finally, in the ad distribution system phase, there is a compound process that acquires all of the steps in classifying an unlabeled crawled web document.

In this section we firstly give detailed information on our datasets then we explain our implementation phases in order. Following diagrams visualize the phases in our study.

Figure 4. 1: Data Collection and Preprocessing Phase

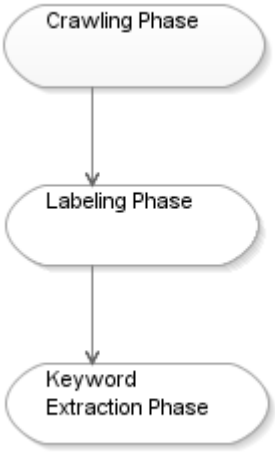


Figure 4. 2: Feature Extraction Phase

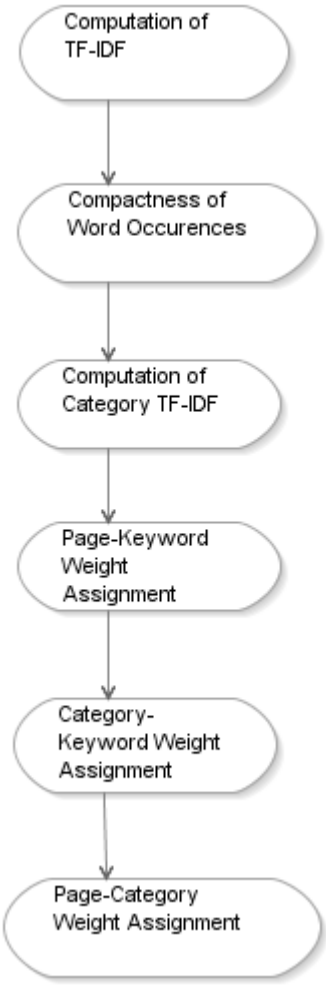
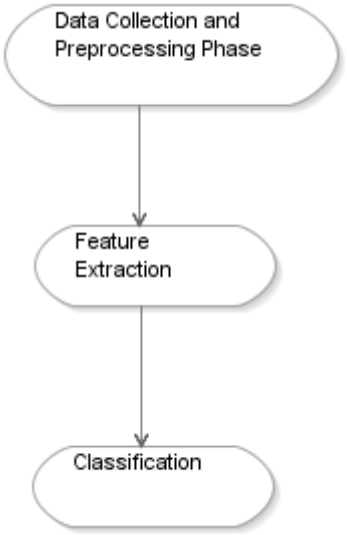


Figure 4. 3: Main Phases



4.1 Data Content

In this section we explain our data sources. We give statistics on our raw data collection and finalized datasets. Information given in this section aims to make our implementation phases more comprehensible.

4.1.1 *Data Sources*

We presented two important considerations in Section 3.1. One of them stated that e-commerce sites dominate the web marketing domain. They are good data sources for categorization of products and product related content. The second consideration is that a classifier designed for a practical ad distribution system should aim to classify any kind of web document. Using these statements as a starting point we decided to form our dataset from Turkish e-commerce sites. In order to keep generality and to train our classifier suitable for any content we tried to use a high number of sites and content pages in these sites. We compromised with the performance issues given in Section 3.2 at a certain level.

The web sites we have used as data sources are as follows: Gittigidiyor.com, Hepsiburada.com, Akakce.com, Tio.com.tr, Koysepete.com, Hemdebufiyata.com, Ereyon.com, Ucuzu.com.

These sites are selected among the most popular e-commerce sites in Turkey. Popularities of these sites are assessed according to Alexa.com Rank parameter given by

Alexa.com. Alexa.com is a widely accepted web statistics source that presents detailed information about web site traffics freely.

4.1.2 *Data Specifications and Datasets*

In this section we will give some statistics about our raw corpus and processed datasets. The following table summarizes general specifications about our raw dataset.

Table 4. 1: Raw Dataset Specifications

Specification	Count
Sites crawled	8
Categories	20
Pages crawled	208146
Labeled pages	199077
All keywords	97615975
All keywords in labeled pages	96933425
All keyword stems	14375
Avg. roots in a page	200
Avg. keywords in a page	1000

In Table 4.1 we give the number of sites we have crawled as 8. There are 20 categories in our dataset. We have crawled 208,146 web pages and determined the category of 199,077 of them to use in the training and test phases of our algorithms. There are a total of 96,933,425 keywords in our labeled pages. These keywords are originated from 14,375 word stems. A web page contains an average of 1,000 keywords and an average of 200 stems.

We have formed different datasets from this raw corpus. We have used different feature extraction approaches. The first approach reduces the number of features to the number of categories and projects weights of all stems as category weights. We formed two datasets using this principle according to different formulas used in the weighting processes. We used another dataset by forming instances directly from stem weights

contained in them. We used different TFIDF limits to select which stems to use. The following table summarizes our datasets. We also used randomly selected 20,000 instance versions of these datasets.

Table 4. 2: Datasets and Specifications

Dataset Name	Method	Samples	Features	Classes
DC1	Category Features + Formula 1	199077	20	20
DC2	Category Features + Formula 2	199077	20	20
DR_0.1	Root Features + TFIDF: 0.1	170325	3836	20
DR_0.2	Root Features + TFIDF: 0.2	160976	1716	20
DC1_20K	Category Features + Formula 1	20000	20	20
DC2_20K	Category Features + Formula 2	20000	20	20
DR_0.1_20K	Root Features + TFIDF: 0.1	20000	3836	20
DR_0.2_20K	Root Features + TFIDF: 0.2	20000	1716	20

We determined the categories of web pages with both automatic and manual processes. Firstly a parser process specialized for each site extracted the label of every crawled page for each site. Usually one or two slightly different parsers were enough for a single site in order to label all of the crawled pages from that site. There were initially 119 categories most of which were closely related semantically. Then we projected all categories to manually selected 20 categories. A few SQL update queries were enough to achieve this in our database layout. Our final categories are given below in Table 4.3.

Table 4. 3: Categories and Samples

Cat. Id	Cat. Name	Samples
1	Telefon-İletişim	2520
2	Takı & Mücevher	10316
3	Sağlık-Bakım-Spor	20625
4	Pet Shop	1862
5	Oyun Hobi	1708
6	Oto Aksesuar	9190
7	Ofis & Kırtasiye	4997
8	Müzik	2011
9	Kitap	5632
10	Hediye ve Çiçek	5602
11	Giyim ve Aksesuar	10239
12	Fotoğraf & Kamera	3659
13	Ev, Dekorasyon & Bahçe	24821
14	Elektronik	18142
15	DVD, Film & TV	1835
16	Doğal Ürünler ve Gıda	2195
17	Bilgisayar	49132
18	Beyaz Eşya & Mutfak	12858
19	Antika & Sanat	305
20	Anne-Bebek	11428

4.2 Data Collection and Preprocessing

We have given the size and content statistics of our datasets in the previous section. We can look further into phases that are used to construct these datasets. In this section, we give detailed information on the collection and preprocessing steps that are used for generating the datasets before weighting and feature extracting phases.

4.2.1 *Crawling Phase*

In the crawling phase we developed a procedure for crawling sites from the Internet. The procedure uses two important property files; AbortRegExp.list and RegExp.list. AbortRegExp.list is a property file for eliminating URLs when crawling. It contains a regular expression list. During the crawling process any URL matching these regular expressions are not crawled. This file tells our process not to crawl file, FTP, mail, Javascript, images or URLs directing to a different scroll on the same page. The content of this file is given in Appendix A. The other property file, RegExp.list, is a site specific regular expression list file. Every line in this list corresponds to a different URL format in a given site. RegExp.list provides a mechanism to limit crawling only within the target site. In Appendix A we give RegExp.list files for each of our sites.

A downloaded web page is stored in the page table in our database (DB) with its type information according to the regular expression its URL matched, its URL, its path in the file system and the site it belongs to. A site that is given to the crawler process is also added to the site table in the DB. We give the DB layout used in our study in Appendix C.

The process is run with the Crawler.sh bash script file whose contents are given in Appendix B.

4.2.2 *Labeling Phase*

The crawler process crawls and forms our physical instances. However, we also need the categories of these samples to form a corpus. In e-commerce sites there are mainly category pages, subcategory pages and product pages. There is usually a single structural format for these page types in a site and category information for every web document is placed in the same place according to that format. A parsing process should get this category information for every web page.

In our crawler process we give different regular expressions for every page type in a site. In this way page type information is stored in the DB. We developed simple parsers for every page type in every site. The number of parsers we developed is not high. There is a maximum of three page types that are necessary for our use in every site and in some sites different page types could be parsed with the same parser.

We developed around 15 similar parsers.

Labeler.sh *bash* script file is used for starting this process. The content of this file is given in Appendix B.

4.2.3 *Keyword Extracting Phase*

Keyword extractor process extracts the keywords in a page, logs information about the location of the keyword in a page and finds stems of the keywords using the Zemberek tool.

Firstly, the process parses a web document in order to get contents of the title tag, *meta* description tag and the *meta* keywords tag. Keywords in these tags are logged with their tag information. Rest of the keywords is also logged in the database (DB). This process also aims to extract sentence structures. Punctuation marks ". ! ?" are taken as sentence ends. Position indexes of a keyword within a sentence and in a document are also stored in the database. One crucial point in our study is that keyword stems are found using the Zemberek tool in this step and stored in the DB. The keyword information is in the page-keyword table in the DB.

All HTML code keywords and punctuation marks are eliminated. Special encoded codes like &amp, &lt, .. are converted to their real values &, <,.. and eliminated. Roots connected with items "ve", "veya", "yada", "&" and "," are also logged to be semantically related, however this information is observed to be noisy and is not used in the following steps of the study.

PageKeywordExtractor.sh bash script file is used for starting this process. The contents of this file is given in Appendix B.

4.3 Feature Extraction

Feature Extraction is the most important phase of our study. We implemented many techniques in order to construct a flexible framework for the classification phase. In this section we give detailed information on the techniques used for feature extraction.

4.3.1 *Feature Extraction Approaches*

4.3.1.1 Computation of TF-IDF

As previously mentioned TF-IDF determine how important a word is in a document and in the whole corpus. TF and IDF parameters are related parameters. They determine importance of a word within a document together.

We find TF of every word stem in each labeled web document. Our procedure simply finds the number of all keywords and the number of occurrences of every word stem in a document. These two numbers are divided and TF of every word stem is found in a web page. This operation is repeated for every labeled page in our data collection.

We also calculate IDF of every word stem in the whole corpus. The number of all pages containing the stem is divided by the number of all documents in the corpus and the logarithm of that number is taken. Resulting number is the IDF of the given word stem.

We defined a new term frequency parameter for categories in our study. Category TF is found simply by dividing the number of occurrences of a word stem in a category by the number of all keywords in that category.

We also defined a new inverse document frequency parameter for categories. Category IDF is found by dividing the number of all categories by the number of categories containing the given word stem and taking the logarithm of that number.

TF-IDF information between word stems and pages are stored in term_freq table and category TF-IDF information are stored in term_freq_cat table in our DB.

4.3.1.2 Compactness of Word Occurrences

TF and IDF parameters are related to the frequency of a word within a document. Although frequency related parameters give us useful information, they do not fully express the importance of a term in a given text. Distributional features are stated to be good alternatives to enhance feature extracting processes based on term frequencies. Distributional features define words by expressing their distribution in a text document.

Experiments show that in contrast to using the traditional term frequency features solely, including the distributional features requires only a little additional cost,

while the categorization performance can be significantly improved [2]. Distributional features we used in our study are compactness of the occurrences of a word and position of the first appearance of the word within a document. We used the position of the first appearance of the word in our weighting processes.

Compactness of the occurrences of a word expresses whether a word appears in a partial zone in a document or it is spread over various parts of the document. If a word appears in many different parts of a document this means it is more related to the content of the document. In other words if the compactness of the occurrences of a word is less then the semantic importance of the word is higher. Mathematical formulation of the compactness we defined and used in our study is as follows:

$$centroid(w) = \frac{\sum_{i=1}^{n} position(w_i)}{n} \qquad (11)$$

where n is the number of occurrences of word w in document d.

$$compactness(w) = \frac{\sum_{i=0}^{n} (position(w_i) - centroid(w))}{n} \qquad (12)$$

where $position(w_i)$ gives the position index of word i in document d.
Compactness information is stored in the term_freq table of our DB.

4.3.1.3 Page-Keyword Weight Assignment

We have evaluated TF, IDF and compactness values of word stems for a web document and category TF and category IDF values of word stems for our dataset classes. In our page-keyword weight assigning process we combine these values with additional weighting parameters coming from web document characteristics, Turkish language characteristics and different distributional feature considerations explained in this section. Having all these considerations our weighting process finds weight values for word stems in each document they occur in. And these weight values are stored in term_freq table in our DB.

There are two additional weighting parameters in our page-keyword

weighting process. One of them is page position coefficient. Keywords occurring in the title, meta description and meta keyword tags are assigned with higher coefficient values in the weighting formulas. Below you can see a table showing these coefficients.

Table 4. 4: Page positions and their coefficients

Page Position	Coefficient
Title Tag	10
Meta Description Tag	6
Meta Keywords Tag	8
Rest of the Page	1

Values in Table 4.4 are set manually.

The other additional weighting parameter is first appearance parameter. This parameter is designed using position of the first occurrence parameter given in distributional features and some well-known Turkish characteristics. Position of the first occurrence parameter defined as a distributional feature [2] is not well suited for web documents in its original form. Web pages are structured differently than normal text documents. Words appearing in document title or meta tags would be always advantageous with the original definition. In our study we have already given special treatment for these tags. However, if we redefine first occurrence parameter within a sentence this can be more helpful. We used the coefficient value 2 for the words in the first two occurrences of our extracted sentences.

For our page-keyword extracting process we have defined 7 parameters so far. These are page-stem TF, page-stem IDF, category-stem TF, category-stem IDF, page-stem compactness, page-stem position and sentence-keyword first appearance. Below, we give the mathematical formulation we used to combine these parameters.

$$W_{kj}(r_k) = \frac{TFIDF_{kj}}{compactness_{kj}} \sum_{i=1}^{n} PC(r_{ki}) * FA(r_{ki}) \qquad (13)$$

Here W_{kj} is defined as weight of root k in document j. $TFIDF_{kj}$ and $compactness_{kj}$ are the previously calculated values for the given root and the document. Root k has n occurrences in document j. We sum the product of position coefficients, PC, and first appearance coefficients, FA, for each occurrence of root k in document j.

4.3.1.4 Category-Keyword Weight Assignment

We defined a similar weight value of keywords for categories. Category-Keyword weights give an intuition of how occurrence of a keyword in a page directs the semantic category of that page to a given category.

$$Wc_{kj}(r_k) = TFIDF_{kj} * \sum_{i=0}^{n} W_{ki}(r_k) \qquad (14)$$

Wc_{kj} is the weight of root k in category j. $TFIDF_{kj}$ is the category TFIDF of root k in document j. $W_{ki}(r_k)$ is weight of root k in document i. There are n documents in category j. For all documents in category j we sum page weights of root k then the total value is multiplied by the previously calculated category TFIDF parameter.

Data calculated by this process is stored in term_freq_cat table of our DB.

4.3.1.5 Page-Category Weight Assignment

Our weighting process passed through many evaluation steps up to this point. We embedded detailed information about distribution of word stems in our corpus among pages and categories in our weight tables. We need a final process to extract a valid dataset from this information. Page-Category weight assigning process finds total category weights of word stems in a page for each category. In this way every page is expressed as 20 weight values corresponding to each category in our dataset. Finally, we can call these weight values our features. We used two slightly different formulas in combining the category weights of word stems in a page.

Formula 1:

$$Wpc_{ij}(p_i) = \sum_{r_k \in p_i}^{k} Wc_{kj}(r_k) \qquad (15)$$

$$p_i \in c_j$$

$Wpc_{ij}(p_i)$ is the weight of category j in page i. $Wc_{kj}(r_k)$ is the weight of root k for category j. Weight of category j in page i is found by summation of category j weights of all roots contained in page i.

Formula 2:

$$Wpc_{ij}(p_i) = \sum_{r_k \in p_i}^{k} Wc_{kj}(r_k) * W_{ki}(r_k) \qquad (16)$$

In Formula 2, $Wpc_{ij}(p_i)$ is again the weight of category j in page i. $Wc_{kj}(r_k)$ is the weight of root k for category j. $W_{ki}(r_k)$ is the weight of root k for page i. Weight of category j in page i is found by summation of product of category j weights of all roots contained in page i and page weights of each root k in page i.

Formula 1 finds page-category weights for a page p by simply summing the category-keyword weights of all keywords roots in p. Formula 2 finds page-category weights for a page p by summing the products of category-keyword weights and page-keyword weights. Definition of Formula 1 assumes that page-keyword weights of keyword roots are embedded in their category-keyword weights. On the other hand Formula 2 uses page-keyword weights explicitly.

4.4 Classification Approaches

We have processed our data and stored it in a highly accessible format. This enabled us to export our processed data as datasets with different formats and contents. We tried different approaches and run several tests in our classification step. The main methods we have used are:

1. Default classifier,
2. SVM with RBF kernel with class features and root features,

3. SVM with linear kernel with class features and root features,
4. SVM with both kernels on PCA transformed root features dataset.

We used many parameter combinations in the algorithms we used in transformation and classification steps to reach highest accuracy rates. We mainly used LIBSVM and LIBLINEAR tools in the classification step. We run transformation tests with Weka package. We implemented a PCA transformation process and some exporter processes at this stage.

4.4.1 *Default Classifier*

In our feature extraction processes we used a feature extraction method that reduces number of features from number of keyword stems to number of categories for text classification. This process assigns 20 weight values for each category to every document in our dataset. With a simple approach we can define a classifier that takes the feature with the highest weight value as the final class of the given document. We implemented this simple classifier and used it on our class feature datasets.

Our default classifier procedure gets the instances with 20 class features from the page_cat_weight table of our DB and assigns the category feature with highest value as the class of the page.

4.4.2 *SVM on Class Features*

We run RBF kernel and linear kernel SVM algorithms on our class feature datasets. For RBF kernel we used LIBSVM and for linear kernel we used LIBLINEAR.

A sample LIBSVM test is run as follows. We firstly export our data in LIBSVM data format. We can export class features according Formula 1 or Formula 2 defined in Section 4.3.1.5. Exported data is divided into training and test sets. Then the training and test sets are scaled. The classifier is trained using the training set. Finally, the labels of the instances in the test set are predicted and stored in an output file.

We also run tests on our class feature datasets using LIBLINEAR with linear kernel. We conduct similar steps with the LIBSVM case. We store the labels of test instances in a file for further processing.

4.4.3 *SVM on Root Features*

We talked about LIBSVM and LIBLINEAR usage in the previous section. We also run similar steps on our datasets with root features.

We extract our dataset with root features in LIBSVM data format as a text file. In the extraction phase we determine the TF-IDF value to limit the number of roots to take as a feature. Then we apply the LIBSVM and LIBLINEAR steps defined in the previous section.

4.4.4 *SVM, PCA on Root Features*

Our first approach used our own feature reduction technique that reduces the number of features to the number of classes. In this test we used an alternative well-known feature reduction technique that is called PCA. We implemented a PCA transformation algorithm and used this algorithm to reduce our features to different numbers. Then we applied the LIBSVM and LIBLINEAR steps defined in Section 4.4.2.

4.5 Ad Distribution System

We developed an ad distribution system framework by using the procedures defined in this chapter. This system extracts the category features of given crawled pages and classify them. Assigned labels of classified pages are stored in DB and the ad serving process send the related ads when this document is viewed in a browser.

The procedure takes the input path of the crawled pages and extracts their features. It scales and classifies the resulting instances using LIBLINEAR and the model file of the DR_0.1 dataset. This classifier achieves good accuracy in a reasonable running time. Then the procedure determines the categories of crawled pages. When these pages are viewed in a browser the JavaScript codes we have implemented call our add distribution server. Our server sends the valid category information and the related ads are displayed on these pages. Our ad distribution server is implemented using Java Server Pages (JSP) and runs on a Tomcat application server.

CHAPTER 5

TEST RESULTS

In this section we give several results we have obtained in our study.

5.1 Default Classifier

We used our default classifier defined in Section 4.3.1 on our class feature datasets. We did not have high accuracy values with this naïve method. On our dataset with category features formed with Formula 1, the classifier classified 74192 instances correctly over 199077 instances. This corresponds to an accuracy rate of 37.26%. And on our dataset with category features formed with Formula 2 the classifier classified 74839 instances correctly over 199077 instances; this corresponds to an accuracy rate of 37.59%. The accuracy rates we obtained with the default classifier are much lower than the accuracy rates obtained using SVM on class features that is given in the following section. This means that class features express a sample document with a combination of all classes. Class feature with the highest weight value does not determine the real category of the sample document.

5.2 SVM on Class Features

In this phase we run SVM classification procedures on our category feature datasets. We used full versions of our datasets and also 20000 sampled versions. We used reduced versions of our datasets in order to have a valid comparison with the PCA transformation phase. For performance reasons we could apply PCA transformations on a dataset of maximum 20000 instances. Larger dataset transformations failed due to memory limitations.

Table 5. 1: SVM on Class Features Results

Dataset Name	Method	Total Samples	Training Set Samples	Test Set Samples	Accuracy (%)
DC1	LIBSVM, RBF, -c 8192 -g 2	199077	179077	20000	88.67
DC2	LIBSVM, RBF, -c 8192 -g 2	199077	179077	20000	84.725
DC1_20K	LIBSVM, RBF, -c 8192 -g 2	20000	18000	2000	86.6
DC2_20K	LIBSVM, RBF, -c 8192 -g 2	20000	18000	2000	82.4
DC1	LIBLINEAR, -c 2048 -e 0.01	199077	179077	20000	75.955
DC2	LIBLINEAR, -c 2048 -e 0.01	199077	179077	20000	67.28
DC1_20K	LIBLINEAR, -c 16 -e 0.01	20000	18000	2000	73.45
DC2_20K	LIBLINEAR, -c 16 -e 0.01	20000	18000	2000	61.1
DC1_20K	LIBLINEAR, -c 2048 -e 0.01	20000	18000	2000	74.45
DC2_20K	LIBLINEAR, -c 2048 -e 0.01	20000	18000	2000	65.4

For LIBSVM we searched for best parameters by using cross validation. For LIBLINEAR we run several tests with different parameter combinations and pick the successful ones.

As the cost parameter *c* in the training command is assigned higher values the computation time and the accuracy of the results increases. And as the gamma parameter *g* in the command gets higher values, the computation time and the accuracy decrease. We found the initial cost value 8192 and the gamma value 2 by applying cross validation. Increasing the cost parameter would probably give slightly higher accuracy rates however would also increase the computation time. The run resulting with 88.67% accuracy classified 20000 instances in 45 minutes and 31 seconds.

Let us look at the results given in Table 5.1 in depth. Using class features enables us to apply powerful SVM configurations on a large dataset. Class features with Formula 1 give higher accuracy rates. Datasets with higher number of samples train the classifier better. The classifier can model the dataset with more detail and can extract more rules when the number of features is higher. SVM with linear kernel is not good at classifying category feature datasets. LIBLINEAR is not as successful as LIBSVM when the number of features is low [10].

5.3 SVM on Root Features

In this phase we run SVM classifications on our root feature datasets. We used 20000 sampled versions of our datasets in this phase again with the same reasons we stated in the previous section. Reduction of the dimension of a feature space decreases the success rates in a classification approach. Therefore, in order to be able to test PCA against our manual feature reduction method, class features, we have to find a classification procedure that uses root features of 20000 instances and is more successful than class feature classifications given in the previous section.

Table 5. 2: SVM on Root Features Results

Dataset Name	Method	TFIDF	Total Samples	Training Set Samples	Test Set Samples	Accuracy (%)
DR_0.1	LIBLINEAR, -c 2048 -e 0.01	0.1	170325	150325	20000	97.7
DR_0.1	LIBLINEAR, -c 64 -e 0.01	0.1	170325	150325	20000	96.58
DR_0.1	LIBLINEAR, -c 16 -e 0.01	0.1	170325	150325	20000	95.155
DR_0.2	LIBLINEAR, -c 16 -e 0.01	0.2	160976	140976	20000	81.46
DR_0.1_20K	LIBLINEAR, -c 16 -e 0.01	0.1	20000	18000	2000	91.3
DR_0.1_20K	LIBLINEAR, -c 64 -e 0.01	0.1	20000	18000	2000	92.9
DR_0.1_20K	LIBLINEAR, -c 2048 -e 0.01	0.1	20000	18000	2000	94.35
DR_0.2_20K	LIBLINEAR, -c 4 -e 0.01	0.2	160976	140976	20000	78.905
DR_0.2_20K	LIBLINEAR, -c 4 -e 0.1	0.2	160976	140976	20000	78.12
DR_0.2_20K	LIBLINEAR, -c 1 -e 0.001	0.2	160976	140976	20000	74.09

We could not apply LIBSVM with RBF kernel on root features due to performance limitations. However, we reached highest accuracy rates using LIBLINEAR linear kernel SVM classifiers. We obtained the best accuracy rate as 97.7 in our study using root features with the linear kernel. The 20000 sampled version of the dataset also reached an accuracy rate of 94.35% with the cost parameter set to 2048. Computation times of most successful linear kernel classification approaches are given in Table 5.3.

Table 5. 3: SVM on Root Features Elapsed Times

Dataset Name	Method	Accuracy	Time Elapsed
DR_0.1	LIBLINEAR, -c 2048 -e 0.01	97.7%	78 min. 12 sec.
DR_0.1	LIBLINEAR, -c 64 -e 0.01	96.58%	13 min. 58 sec.
DR_0.1	LIBLINEAR, -c 16 -e 0.01	95.155%	5 min. 41 sec.

Table 5.3 presents that SVM with linear kernel on root features gives the highest accuracy rates in lowest elapsed times in our study. Root features express text documents with higher detail than class features. In this way they provide the learning phase of the classification scheme with more elaborate rules related to distribution of the samples within the dataset. In addition, SVM with linear kernel is suitable for classification in high dimensional feature spaces. Linear kernel learning algorithm can construct accurate classification models in shorter time compared to the RBF kernel.

5.4 SVM, PCA on Root Features

We applied PCA transformation on our most successful 20K dataset with root features, which is DR_0.1_20K. We reduced number of features using PCA method to different numbers and tested our classification methods on the resulting feature spaces.

Table 5. 4: SVM on PCA Transformed Root Features Results

# of features	Method	Accuracy (%)
20	LIBSVM, RBF, -c 8192 -g 2	29.45
20	LIBSVM, RBF, -c 16 -g 0.05	26.8
20	LIBSVM, RBF, -c 1 -g 0.05	26.8
200	LIBSVM, RBF, -c 8192 -g 2	19.65
200	LIBSVM, RBF, -c 16 -g 0.05	26.3
200	LIBSVM, RBF, -c 1 -g 0.05	26.3
2000	LIBSVM, RBF, -c 8192 -g 2	17.85
20	LIBLINEAR, -c 16 -e 0.01	4.5
200	LIBLINEAR, -c 16 -e 0.01	10.45

Classification tasks we run on the PCA transformed versions of DR_0.1_20K dataset did not give satisfactory accuracy values. We obtained the best result, 29.45%, using RBF kernel and PCA 20 feature dataset. This shows that PCA transformation lose most of the data charactheristics contained in the root feature dataset. PCA transformation is a common mathematical method in the literature. The method is not specific to any domain or dataset. On the other hand the class feature reduction method employs text classification related schemes like TF-IDF and use weighting formulations specific to our data collection. Data specific properties of the class features enable the method to give much better accuracy rates than the PCA method.

5.5 Distinct Site Tests

Having determined the most successful classification schemes, we measure success rate of our approaches in classifying totally new sites in test sets. We formed our training and test sets with web documents from different web sites. Then we applied SVM with RBF kernel on class features and SVM with linear kernel on root features. Table 5.5 presents our approaches and results. Our training set is composed of web documents from sites: Hepsiburada.com, Akakce.com, Tio.com.tr, Koysepete.com, Hemdebufiyata.com, Ucuzu.com. And our test set consists of web documents from sites: Gittigidiyor.com, Ereyon.com.tr.

Table 5. 5: Distinct Site Tests

Method	TFIDF	Total Samples	Training Set Samples	Test Set Samples	Accuracy (%)
LIBLINEAR, -c 16 -e 0.01	0.1	170325	148039	22286	44.23
LIBLINEAR, -c 2048 -e 0.01	0.1	170325	148039	22286	31.97
LIBSVM, -c 8192 -g 2	-	199077	176791	22286	3.84
LIBSVM, -c 16 -g 2	-	199077	176791	22286	7.86

These results show that our training dataset should be generalized in order to make our classification approaches usable in a practical contextual advertising

system. The class feature method gave low accuracy rates and was unsuccessful in classifying web documents from different web sites. The root feature method gives higher accuracy values and seems to be more adaptive in classifying different web document structures. However a more generalized training set should be used with both approaches.

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

In this thesis we researched various feature extraction techniques for text classification and combined these techniques for designing a classifier specific to the Turkish marketing domain. We also tested several SVM classification techniques on datasets in different feature spaces. Additionally, we designed and implemented a contextual ad distribution system combining the phases in our study.

Weighting schemes defined in the feature extraction phase is the most crucial step in this study. We used TFIDF, compactness as a distributional feature and position index within a sentence as a modified distributional feature from a previous work done in information extraction area. We enhanced these methods with weighting definitions for different HTML tags and our weighting formulas for web pages and roots. We used a different feature extraction technique that reduces the number of features to the number of classes for text categorization. In this part of our study we contributed with new TFIDF definitions for classes and new weighting formulas that relate roots and category features.

After a detailed feature extraction phase we tried several SVM classification methods listed in Chapter 5. We conclude that a dataset consisting of text documents is best expressed with root feature methods. Feature reduction methods built on root features reduce the performance in terms of accuracy and speed. SVM with linear kernel on root features is the best classification technique in terms of accuracy and speed for web documents.

Additionally, the category feature extraction method built on root feature extraction method is a successful scheme for limiting the number of features in text classification. Category features can keep the semantic content expressed by root features at a high level. Datasets consisting of category features are classified better with SVMs with RBF kernel. The procedure, as a whole with the feature extraction and classification phases, reaches an adequate level of success in terms of speed and accuracy.

Moreover, the method gives far more accurate results than the standard feature reduction method PCA.

SVM with linear kernel on root features and SVM with RBF kernel on category features are two successful classification methods. SVM with linear kernel on root features gives much better performance on our current dataset. However, our classification methods may have lower success rates on web documents with different structures and contents. Characteristics of web pages in e-commerce sites may not express all web content at the same level. For this reason we may need to generalize our training set and widen its size according to the classification task we have. Root features scheme may suffer with larger datasets since larger dataset size means indefinitely larger number of features. Larger number of features will increase the task completion time. There is a trade off between higher accuracy and shorter processing time. Category features method is more scalable with the dataset size since number of features does not increase with dataset size.

Our conclusions in this thesis can be summarized as follows:

1. Special feature reduction methods can perform far better than standardized methods like PCA.
2. Statistical feature extraction techniques enhanced with distributional features give semantically accurate results on web text documents.
3. SVM with linear kernel function gives best performance in terms of accuracy and speed on text documents expressed as keyword root features.

Our study in this thesis can be extended in two main areas. One of them is to improve the feature extraction and classification phases. The other improvement is the design and development of the ad distribution system as a practical mechanism.

In the feature extraction phase semantic and language processing techniques can be added. A Turkish WordNet structure can be used to improve weighting procedures. Named entity recognizers, part of speech taggers and language processing techniques to extract sentence structures can be used.

In the classification phase the comparison between SVM with linear kernel on root features and SVM with RBF kernel on category features can be further analyzed. Scalability of root feature method can be tested with larger dataset sizes and the performance border between two methods can be searched.

On the ad distribution system side, our dataset can be generalized with new labeled web documents that have different page structures and contents. A more general training set can better express any kind of content on the web. In order to do this new labeled content from different web site categories should be supplied or created as we did in our labeling phase. Our current crawling, labeling and keyword extracting procedures should be made more efficient to build a practical contextual ad distribution system.

Finally, our datasets can be used in further studies. We have weight values of thousands of words according to a category set. These weight values can be used to define a relatedness measure among the word stems. The category set can also be extended, a hierarchy of categories can be formed. A semantic network of several levels can be built. A study on this path can lead to a Turkish Wordnet framework.

REFERENCES

[1] Yıldız, H.K.; Gençtav, M.; Usta, N.; Diri, B.; Amasyalı, M.F., A New Feature Extraction Method for Text Classification, Signal Processing and Communications Applications, SIU 2007. IEEE 15th Volume , Issue , 11-13 June 2007 pp. 1-4, 2007

[2] Xiao-Bing Xue and Zhi-Hua Zhou, Distributional Features for Text Categorization, 17th European conference on machine learning, Berlin, Germany, September 18-22, 2006

[3] Roberto Basili and Alessandro Moschitti, A robust model for text classification, Tools with Artificial Intelligence, Proceedings of the 13th International Conference, Dallas, TX, USA, November, 2001

[4] Thorsten Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features, Proceedings of ECML-98, 10th European Conference on Machine Learning, 1998

[5] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," Machine Learning, vol. V39, no. 2, pp. 103-134, May 2000

[6] Alaa Abi-Haidar, Jasleen Kaur1, Ana Maguitman, Predrag Radivojac, Andreas Rechtsteiner, Karin Verspoor, Zhiping Wang and Luis M. Rocha, Uncovering protein interaction in abstracts and text using a novel linear model and word proximity networks, Genome Biology 2008, 9(Suppl 2):S11, 2008

[7] Princeton University, WordNet a lexical database for the English Language, http://wordnet.princeton.edu/, 2006

[8] Orhan Bilgin, Özlem Çetinoğlu, Kemal Oflazer, Building a WordNet for Turkish, Romanian Journal of Information Science and Technology Volume 7, Numbers 1-2, pp. 163-172, 2004

[9] D. I. Moldovan and R. Mihalcea. Using WordNet and lexical operators to improve Internet searches. IEEE Internet Computing, January-February, 2000

[10] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, A Practical Guide to Support Vector Classification, 2003

[11] Richard O. Duda, Peter E. Hart, David G. Stork, Pattern Classification, 2nd Edition, pp. 259-272, 2000

[12] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001

[13] Chih-Chung Chang and Chih-Jen Lin, LIBLINEAR : a library for large linear classification, http://www.csie.ntu.edu.tw/~cjlin/liblinear, 2008

[14] Ian H. Witten and Eibe Frank (2005) "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[15] MathWorks, NIST, JAMA : A Java Matrix Packafe, http://math.nist.gov/javanumerics/jama/, 2005

[16] Ahmet Afşın Akın, Mehmet Dündar Akın, Zemberek, http://code.google.com/p/zemberek/, 2008

[17] CellarStone Inc. (2006), http://www.qcommission.com/salescommission_details.htm, QCommission.com, retrieved June 25, 2007

[18] Marios Alexandrou (February 4th, 2007), CPM vs. CPC vs. CPA, All Things SEM, retrieved November 11, 2007

[19] Interactive Advertising Bureau, http://www.iab.net/, IAB - Dedicated to the Continued Growth of the Interactive Advertising Marketplace, 2008

[20] Kenny, D. and Marshall, J. (November-December 2000). "Contextual Marketing: The Real Business of the Internet". Harvard Business Review. http://hbswk.hbs.edu/archive/2124.html. Retrieved on 22 July 2008.

[21] Huma Lodhi, Craig Saunders, Nello Cristianini, Chris Watkins, Bernhard Scholkopf, Text classification using string kernels, Journal of Machine Learning Research, pp. 563-569, 2002

[22] Wall, Michael E., Andreas Rechtsteiner, Luis M. Rocha. "Singular value decomposition and principal component analysis". in A Practical Approach to Microarray Data Analysis. D.P. Berrar, W. Dubitzky, M. Granzow, eds. pp. 91-109, 2003

[23] S. Leach. Singular value decomposition - a primer. Unpublished Manuscript, Department of Computer Science, Brown University, Providence, RI, USA.

[24] Juan Ramos, Using TF-IDF to Determine Word Relevance in Document Queries, Instructional Conference on Machine Learning Web Site and Instructions, 2003

APPENDIX A

PROPERTY FILES

Property Files

Database.properties

properties file

database.driver=org.gjt.mm.mysql.Driver

database.url=jdbc:mysql://localhost:3306/thesis?useUnicode=true&characterEncoding=UTF-8

database.username=root

database.password=xxxxx123

AbortRegExp.list

#skip special hrefs

AB_1=(?s).*(file|ftp|mailto|javascript):(?s).*

#skip image etc. hrefs

AB_2=(?s).*.(gif|GIF|jpg|JPG|png|PNG|ico|ICO|css|sit|eps|wmf|zip|ppt|mpg|xls|gz|rpm|tgz|mov|MOV|exe|jpeg|JPEG|bmp|BMP)(?s).*

#skip query like hrefs

#AB_3=[?*!@=]

AB_3=(?s).*#(?s).*

RegExp.list Files

Hemdebufiyata.com.tr

HDBF_SUB_KAT=http://www.hemdebufiyata.com.tr/BrowseProducts(_)?.aspx\?DepartmentID=[0-9]+

HDBF_ANA_KAT=http://www.hemdebufiyata.com.tr/.+.aspx\?DepartmentId=[0-9]+

HDBF_URUN=http://www.hemdebufiyata.com.tr/ShowProduct.aspx\?ID=[0-9]+

HDBF_DEF=http://www.hemdebufiyata.com.tr/Default_.aspx

Akakce.com

AKK_KAT=http://www.akakce.com/[-a-zA-Z0-9]+.(html|asp)

AKK_KAT_NUM=http://www.akakce.com/[-a-zA-Z0-9]+(,[-a-zA-Z0-9]{1,2})+.(html|asp)

AKK_URUN=http://www.akakce.com/[-a-zA-Z0-9]+/en-ucuz-[-a-zA-Z0-9]+-fiyati,[0-9]+.(html|asp)

Ereyon.com.tr

ERYN_ANA_KAT=http://www.ereyon.com.tr/KMdefault.aspx\?ctlid=[0-9]+(&ix=[0-9]+)?

ERYN_SUB_KAT=http://www.ereyon.com.tr/KMdefault.aspx\?ctrid=[0-9]+(&ix=[0-9]+)?

ERYN_URUN=http://www.ereyon.com.tr/store/ProductDetails.aspx\?productId=[0-9]+

Gittigidiyor.com

GTGD_URUN=http://urun.gittigidiyor.com/[a-zA-Z0-9-\?=&_.]+

GTGD_URUN=http://www.gittigidiyor.com/php/urun.php\?id=[0-9]+

GTGD_ANA_KAT=http://[a-zA-Z]+.gittigidiyor.com(/)?

GTGD_SUB_KAT=http://[a-vx-zA-VX-Z-]+.gittigidiyor.com/[a-zA-Z-_]+

Hepsiburada.com

HB_ANA_KAT=http://www.hepsiburada.com/department.aspx\?catid=[0-9]{1,5}

HB_SUB_KAT=http://www.hepsiburada.com/department.aspx\?CategoryID=[a-zA-Z0-9-\?=&_.\|]+

HB_URUN=http://www.hepsiburada.com/[a-zA-Z0-9-/\?=&_.]*productdetails.aspx\?categoryid=[a-zA-Z0-9-\?=&_.]+productid=[a-zA-Z0-9-\?=&_.]+

HB_DEF=http://www.hepsiburada.com/.*

Koysepete.com

KSP_KAT=http://www.koysepete.com/en\+ucuz\+.+\+[0-9]+\+[0-9]+\+listesi.aspx

KSP_URUN=http://www.koysepete.com/en\+ucuz\+.+\+[0-9]+.aspx

Nevaria.com

NVR_ANA_KAT=http://www.nevaria.com/kategoriler.asp\?CatID=[0-9]+

NVR_SUB_KAT=http://www.nevaria.com/arama_sonuc.asp\?cmd=search&CatID=[0-9]+

NVR_URUN=http://www.nevaria.com/detay.asp\?ID=[0-9]+

Tio.com.tr

TIO_URUN=http://www.tio.com.tr/.+/(UrunDetay|UrunBilgileri|UrunYorum).aspx\?id=.+

TIO_ANA_KAT=http://www.tio.com.tr/Kategori.aspx\?cid=[0-9]+

TIO_SUB_KAT_1=http://www.tio.com.tr/Kategori.aspx\?rcid=[0-9]+&cid=[0-9]+(&cn=.+)?

TIO_SUB_KAT_2=http://www.tio.com.tr/.+/Urunler.aspx\?cid=[0-9]+

TIO_SUB_KAT_3=http://www.tio.com.tr/.+/urunliste.aspx\?cid=[0-9]+

#TIO_DEF=http://www.tio.com.tr/.*

Ucuzu.com

UCZ_KAT=http://www.ucuzu.com/.+/(0-.+.html)?

UCZ_URUN=http://www.ucuzu.com/.+/.+.html

APPENDIX B

BASH SCRIPTS

Crawler.sh

```
cd /srv/thesis/thesis

nohup java -jar crawler.jar http://www.hemdebufiyata.com.tr/ >& Crawler.hdbf.log &
```

DefClassifier.sh

```
cd /srv/thesis/thesis

nohup java -jar DefClassifier.jar 2 > DefClassifier.2.log &
```

ExtractFeatures.sh

```
nohup java -Xmx1024M -jar ExtractFeatures.jar combine btrns-1230055216862.txt dat-c-20K > ExtractFeatures.com.log &
```

Labeler.sh

```
cd /srv/thesis/thesis

nohup java -jar labeler.jar /srv/thesis/thesis/crawledsites/hemdebufiyata >& Labeler.hemdebufiyata.log &
```

PageKeywordExtractor.sh

```
cd /srv/thesis/thesis

nohup java -jar PageKeywordExtractor.jar /srv/thesis/thesis/crawledsites/hemdebufiyata >& PageKeywordExtractor.hdbf.log &
```

PCAcorr.sh

```
cd /srv/thesis/thesis

nohup java -Xmx2560M -jar PCAcorr.jar dat-f-20K  >& PCAcorr.20K.log &
```

TFFinder.sh

```
cd /srv/thesis/thesis

nohup java -jar TFFinder.jar idf >& TFFinder.idf.log
```

APPENDIX C

DATABASE SQLs

```
create table site(
SITEID int(11) auto_increment,
NAME varchar(60),
URL varchar(255),
primary key (SITEID)
);
create table page(
PAGEID int(11) auto_increment,
SITEID int(11),
URL varchar(255),
PPATH varchar(255),
CATEGORY varchar(20),
LABEL varchar(255),
FINAL_LABEL varchar(255) collate utf8_turkish_ci,
primary key (PAGEID),
foreign key (SITEID) references site(SITEID)
);
ALTER TABLE page CONVERT TO CHARACTER SET utf8 COLLATE
utf8_turkish_ci;

create table root(
ROOTID int(11) auto_increment,
ROOT varchar(255),
primary key(ROOTID)
);
```

```
ALTER TABLE root CONVERT TO CHARACTER SET utf8 COLLATE utf8_turkish_ci;
create table page_keyword(
PAGE_KEYWORDID int(11) auto_increment,
PAGEID int(11),
KEYWORD varchar(255),
ROOTID int(11),
IMPORTANCE int,
SENTENCE varchar(255),
PAGEPART varchar(60),
POSITION int,
SENTENCE_POS int,
primary key (PAGE_KEYWORDID),
foreign key (PAGEID) references page(PAGEID),
foreign key (ROOTID) references root(ROOTID)
);

ALTER TABLE page_keyword CONVERT TO CHARACTER SET utf8 COLLATE utf8_turkish_ci;

create table term_freq(
TFID int(11) auto_increment,
PAGEID int(11),
ROOTID int(11),
TF float,
COMPACTNESS float,
IDF float,
WEIGHT double;
foreign key (PAGEID) references page(PAGEID),
foreign key (ROOTID) references root(ROOTID),
primary key(TFID)
);
```

```
create table term_freq_cat(
TFID int(11) auto_increment,
CATID int(11),
ROOTID int(11),
TF float,
COMPACTNESS float,
idf float,
WEIGHT double,
foreign key (CATID) references categrory(CATID),
foreign key (ROOTID) references root(ROOTID),
primary key(TFID)
);

create table category(
CATID int(11) auto_increment,
CATEGORY varchar(60) collate utf8_turkish_ci,
primary key(CATID)
)


create table related_roots(
LROOTID int(11),
RROOTID int(11),
primary key (LROOTID,RROOTID),
foreign key (LROOTID) references root(ROOTID),
foreign key (RROOTID) references root(ROOTID)
);
```

```
create table page_cat_weight(
PCWID int(11) auto_increment,
PAGEID int(11),
CATID int(11),
WEIGHT double,
TYPE int,
foreign key (PAGEID) references page(PAGEID),
foreign key (CATID) references category(CATID),
primary key (PCWID)
)
CREATE INDEX tf_root ON term_freq(ROOTID);
CREATE INDEX tf_page ON term_freq(PAGEID);
CREATE INDEX pk_root ON page_keyword(ROOTID);
CREATE INDEX pk_page ON page_keyword(PAGEID);

create table page_categorized(
CZID int(11) AUTO_INCREMENT,
PAGEID int(11),
CATID int(11),
foreign key (PAGEID) references tc_page(PAGEID),
foreign key (CATID) references category(CATID),
primary key (CZID)
)
create table ad(
ADID int(11),
TARGETURL varchar2(255),
primary key(ADID)
)
create table ad_category(
```

```
ACID int(11),
ADID int(11),
CATID int(11),
foreign key (ADID) references ad(ADID),
foreign key (CATID) references category(CATID),
primary key (ACID)
)
create table tc_page(
PAGEID int(11) auto_increment,
SITEID int(11),
URL varchar(255),
PPATH varchar(255),
CATEGORY varchar(20),
LABEL varchar(255),
FINAL_LABEL varchar(255) collate utf8_turkish_ci,
primary key (PAGEID),
foreign key (SITEID) references site(SITEID)
);
ALTER TABLE tc_page CONVERT TO CHARACTER SET utf8 COLLATE
utf8_turkish_ci;
create table tc_page_keyword(
PAGE_KEYWORDID int(11) auto_increment,
PAGEID int(11),
KEYWORD varchar(255),
ROOTID int(11),
IMPORTANCE int,
SENTENCE varchar(255),
PAGEPART varchar(60),
POSITION int,
SENTENCE_POS int,
primary key (PAGE_KEYWORDID),
```

```
foreign key (PAGEID) references tc_page(PAGEID),
foreign key (ROOTID) references root(ROOTID)
);

ALTER TABLE tc_page_keyword CONVERT TO CHARACTER SET utf8
COLLATE utf8_turkish_ci;

create table tc_term_freq(
TFID int(11) auto_increment,
PAGEID int(11),
ROOTID int(11),
TF float,
COMPACTNESS float,
IDF float,
WEIGHT double,
foreign key (PAGEID) references tc_page(PAGEID),
foreign key (ROOTID) references root(ROOTID),
primary key(TFID)
);

create table tc_page_cat_weight(
PCWID int(11) auto_increment,
PAGEID int(11),
CATID int(11),
WEIGHT double,
TYPE int,
foreign key (PAGEID) references tc_page(PAGEID),
foreign key (CATID) references category(CATID),
primary key (PCWID)
);
```

APPENDIX D

TOOL COMMANDS

LIBSVM Commands

A sample classification task:

>./subset.py ../../thesis/com-1230146036012.txt 2000 pca-2000-test pca-2000-train

We use *subset.py* to split a dataset into training and test sets. The above command splits the "../../thesis/com-1230146036012.txt" dataset file as 2000 instances in the "pca-2000-test" file and rest in the "pca-2000-train" file.

>./svm-scale -l -1 -u 1 -s fe-1-20000-range tools/fe-1-20000-train > fe-1-20000-train.scale

>./svm-scale -r fe-1-20000-range tools/fe-1-20000-test > fe-1-20000-test.scale

We use *svm-scale* command to scale features in training and test sets. First command scales "tools/fe-1-20000-train" training file. Scaled information is written into "fe-1-20000-train.scale" file. And the scaling information is stored in "fe-1-20000-range" later to be used when scaling the test file with the same ratio. And in the second command "tools/fe-1-20000-test" is scaled with the information in "fe-1-20000-range". Scaled data is ready in the "fe-1-20000-test.scale" file.

./svm-train -s 0 -t 2 -c 8192 -g 2 tools/fe-1-1000-train.scale tools/fe-1-1000-train.model

After the scaling we use *svm-train* command to build a model file and extract classification rules. In the above command -s 0 states we will use C-SVC [12] as SVM type, -t 2 means we will use RBF kernel. -c 8192 sets our cost parameter and -g 2 sets the gamma parameter in the RBF. Our training dataset is "tools/fe-1-1000-train.scale"

and our trained model is stored in "tools/fe-1-1000-train.model" file. Most of our classification work is done when we obtain the training model file.

> ./easy.py <train file> <test file>

The *easy.py* command gives the optimum c and g parameters for a given dataset.

>./svm-predict tools/pca-20-test pca-20.model predict-pca-20.output

Finally, we predict the labels of instances in our test set with the *svm-predict* command. With this command, instances in the "tools/pca-20-test" file are classified according to the "pca-20.model" file obtained in the previous step. Predicted labels are given in "predict-pca-20.output" file.

LIBLINEAR Commands

A sample classification task:

>./train -s 4 -c 16 -e 0.01 rf-full-1712-train rf-full-1712.model

We use *train* command to learn the classification rules from the training set. In the above command we use 16 as cost parameter and 0.01 as tolerance of the stopping criterion [13].

>./predict rf-full-1712-test rf-full-1712.model rf-full-1712.output

We predict the labels of instances in the test set with the *predict* command. Labels of test instances are stored in *rf-full-1712.output* text file.