MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Finding People on the Internet

MASTER THESIS

Juraj Jurčo

spring 2012

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

In Brno, May 28, 2012
Juraj Jurčo

Prehlásenie

Prehlasujem, že táto diplomová práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

V Brne, 28. máj, 2012
Juraj Jurčo

Advisor: doc. RNDr. Lubomír Popelínský

Acknowledgement

First and foremost, I would like to thank my advisor Lubomír Popelínský for his everlasting support and friendly guidance, this work would have never been completed without his patience and constant inspiration.

I am also grateful to several other people for their support during work on this thesis – I would like to thank Pavel Brázdil for raising important questions and motivation during my intership in Porto, Martin Víta for his help and mainly for suggesting this great topic. I would like to express my gratitude to my parents and grandparents for their invaluable and loving support during my studies at university and the trust they have put in me. Last but not least I am much obliged to my very good friend Ľubica Backová for her textual and grammar corrections.

Poďakovanie

Predovšetkým by som chcel poďakovať vedúcemu mojej práce Lubomírovi Popelínskému za jeho podporu a priateľský prístup. Táto práca by nikdy nevznikla bez jeho trpezlivosti a neustálej inšpirácie.

Som takisto vďačný niekoľkým ďalším ľuďom – chcem poďakovať Pavlovi Brázdilovi za dôležité otázky a motiváciu počas môjho pobytu v Porte, Martinovi Vítovi za jeho pomoc a pretože táto téma bola jeho skvelý nápad. Hlavne by som chcel poďakovať mojim rodičom a starým rodičom za neoceniteľnú a láskavú podporu môjho štúdia na univerzite a dôveru, ktorú mi venovali.

Napokon by som chcel poďakovať mojej veľmi dobrej kamarátke Ľubici Backovej za jej textové a gramatické opravy.

Abstract

In this work we deal with the problem of finding people on the Internet. Our main aim is to find information about people working in the field of informatics as managers or owners of companies who are suitable candidates to reach out to offer a partnership. We describe the problem of *Business partner search* and we propose an algorithm of search by hand which may serve as the basis for real algorithm. Because of the fact the downloaded documents from the Internet usually contain a lot of irrelevant documents, the tests realized in this work are focused on their basic pre-filtering. We found out that decreasing of too high baseline in the training set may lead to better performance provided that we selected relevant documents on unseen data. Furthermore, when we classify documents about managers working in the field of informatics it is comparable to classify them by two stage classifiers.The results of these tests are not only applicable for basic search results pre-filtering but the thesis also provides theoretical background for real application able to search potential business partners.

Abstrakt

V tejto práci sa zaoberáme problémom hľadania ľudí na internete. Naším zámerom je nájsť informácie o osobách pracujúcich v oblasti informatiky na manažérskych pozíciách alebo vlastníkov firiem, ktorí sú vhodnými kandidátmi na spoluprácu. Popisujeme problém *vyhľadávania obchodných partnerov* a navrhujeme algoritmus ručného vyhľadávania, ktorý môže slúžiť ako základ pre reálny algoritmus. Keďže dokumenty stiahnuté z internetu zvyčajne obsahujú veľké množstvo irelevantných dokumentov, testy v tejto práci sú zamerané na ich základné prefiltrovanie. Zistili sme, že zníženie baseline na trénovacej množine môže viesť k lepším výsledkom klasifikácie. Navyše, keď klasifikujeme dokumenty o manažéroch v oblasti informatiky, podobné výsledky môžeme dostať aj dvojstupňovým klasifikovaním. Výsledky tychto testov sú nielen aplikovateľné pre filtrovanie výsledkov vyhľadávania, ale táto práca poskytuje teoreticky zaklad pre reálnu aplikáciu vyhľadávajúcu potencionálnych priemyselnych partnerov.

Keywords

Internet, data mining, web mining, web person search, business partner search, classification, pre-processing

Contents

Introduction

We live in the age of information technologies and the Internet. More and more people are using the Internet every day and everybody leaves their fingerprints of the work there. People present themselves, share their opinions, emotions and personal information in general. The Internet has also become one big market place where businessmen and whole companies offer, sell and distribute their products. With the growing number of information about people on the Internet people search for other people too. They have personal reasons but there are also businessmen and companies searching for suitable candidates to reach out to and offer a partnership.

In the first chapter we introduce reader to the Information retrieval topic. We will talk about the most used information retrieval systems – web search engines and types of queries they support. A short introduction to data mining techniques is in the second chapter. We discuss three key parts of data mining there: *pre-processing*, *data mining* and *validation*. In chapter three we define Web Person Search problem. At first we divided people into two main groups by their presence or absence on the Internet. We described the problem, its difficulties and we made a survey of methods used for person disambiguation. In chapter four we compared the problem of Web Person search with the problem of Business Partner search. The search for business partners is slightly different in comparison to the simple person search, therefore we discussed also other types of problems we have to solve. We also made a survey of useful information resources appropriate for this problem. In chapter five we discussed the techniques of preprocessing which we used with focus on web documents. In chapter seven we describe the tests we performed. We collected, pre-processed and classified gathered documents from the web. The results of classification are described in chapter eight. In the last chapter we make conclusion of the work and describe possible extension and future work.

Chapter 1

Information retrieval

"Sharing knowledge is not about giving people something,
or getting something from them.
That is only valid for information sharing.
Sharing knowledge occurs when people are genuinely interested in
helping one another develop new capacities for action;
it is about creating learning processes."

– Peter M. Senge [33]

Ever since we can remember people gather and exchange data. The inception of sharing information was in sound, speech and gestures. Step by step people found another ways how to share information when they want to avoid misinterpretation. Books and written text or symbols were a good transfer medium between different people and generations of people. Author wrote his thoughts and findings so everyone had the same interpretation. Books play important role in sharing and exchanging knowledge or information over the centuries. People have built libraries to facilitate the exchange of books and therefore the exchange of knowledge.

With the invention of information technology all the gathering, exchanging and sharing of information has become easier. The latest medium for sharing and exchanging information is with no doubt the Internet. From the beginning the Internet was used only for exchanging knowledge between scientists. With the increasing number of people and companies connected to the network, the number of information is rapidly growing. Simplicity and speed of obtaining information from this source has made it a dominant seeking method for today's modern people. Nowadays it is more usual to see people search on the web rather than in a library.

1.1 Query types

Search of a document is provided by *Information Retrieval* (IR) system. User sends query to IR system and it returns a set of documents according to the query. It represents the user's information needs which is in one of the following forms: *Keyword queries*, *Boolean queries*, *Phrase queries*, *Proximity queries*, *Full Document queries* and *Natural Language queries* [20].

Keyword queries The user query is usually composed of few *keywords* also called *terms*. Terms in the query are usually connected with some kind of `AND` operator. So IR system looks for the presence of all words in the query, but not necessarily all of them. For example for the query "*Machine learning*" system translates it like "*Machine* `AND` *learning*" and finds documents according to this query. The resultant documents are ranked and presented for the user. In some IR systems the order of the words also plays an important role.

Boolean queries The query can contain also boolean operators such as `AND`, `OR`, `NOT`. The query is then represented as logical formula and the document is returned by the system only in the case the formula is valid in that document.

Phrase queries This type of query is compound of a phrase or whole sentence. In search engine there is usually a phrase enclosed in double quotes. All returned documents should contain at least one occurrence of this phrase.

Proximity queries The proximity query is a composition of *Phrase queries* and *Keyword queries*. The distance between terms and phrases is used to count relevance of the document. When they occur close to each other, the document is more relevant for the user.

Full Document queries These are queries where the user posts the whole document. IR system then searches for similar documents and returns them sorted by relevance to the user.

Natural Language queries Natural Language query is a type of query where user can specify his or her needs in natural language. It is the ideal state, but this type of queries are not fully supported by current IR systems.

Some systems try to handle this kind of queries only partially. They focus only on some subset of queries, e.g. requesting for some services or definitions.

1.2 Web search engines

The most used and known Information retrieval systems are web search engines. They are especially designed to search for information available on the Internet. They usually search for content of the Internet and they make their own database of documents but collaboration between search engines exists as well. The most famous and used search engine is with no doubt *Google* with 91.06% then *bing* with 3.58%, *Yahoo!* with 3.56% and all other with 1.80% [1][1]. There is a cooperation between *bing* and *Yahoo!* [32]. They provide similar results for the same query and obviously also the number of users for both of them is similar.

The success of these search engines is also in the services they provide. They try to handle all types of queries and give users what they are really searching for. To be able to give users a good responses search engines try to implement all kinds of user query forms and their combinations. Search engines also add a new functionality for searching. For example Google is able to search for synonyms or replace $*$ sign with any unknown term. Google also adds a new functionality compared to other search engines. The list of search options for the major three web search engines is in Table 1.2. Like we can see there, all of them implements searching for the first four basic query forms: *Keyword queries*, *Boolean queries*, *Phrase queries* and *Proximity queries*. Google also extends its functionality with the fifth query form *Full document queries*. You can specify in the query the URL of the document and the search engine will search only in documents similar to that specified. Google has recently also added functionality which allows you to search for similar images. For the query refinement you can also use another functionality, for example you can search in specific site, in document's title or URL. Or you can just search in specific types of documents.

However the power of a search engine is not only in the number of covered types of queries. Users are usually expecting fast results. Even skilled search engine users will not use it if it has a slow response. The relevance of the returned documents is also important. Users will prefer to use the search engine which will return to them what they were looking for among the first links,rather than using those which offer the information lost some-

1. Statistics are for the last year, exactly from April 2011 to March 2012

	Google	Yahoo!	Bing
Exactly as it is	+<word>	+<word>	+<word>
Exclude terms	-<word>	-<word>	-<word>
Exact phrase	"<phrase>"	"<phrase>"	"<phrase>"
OR,AND,NOT	<t1> OR <t2>	<t1> OR <t2>	<t1> OR <t2>
Synonyms	~ <word>	Not possible	Not possible
Fill in the blanks	<t1> * <t2>	Not possible	Not possible
Alternate query types			
Search for links	link:<URL>	link:<URL>	link:<URL>
Similar pages	related:<site>	Not possible	Not possible
Query modifiers			
Search in site	site:<URL>	site:<URL>	site:<URL>
Search in title	intitle:<word>	intitle:<word>	intitle:<word>
Search in URL	inurl:<word>	inurl:<word>	inurl:<word>
File type search	filetype:<ext>	filetype:<ext>	filetype:<ext>

Table 1.1: Overview of symbols used by search engines

where between the results. Therefore search engines use various techniques how to increase the precision and the recall of retrieved documents. They usually rank documents and index them by some rules. For the ranking of the importance of the page they usually use PageRank[2] or similar technique. Pages with the higher score are then placed in higher positions. Page creators or spammers sometimes put inappropriate content on pages because they want to raise up their position between results. Therefore search engines use also TrustRank[3] method to penalise such pages.The search engines can return better results for the user if they also remember the search history of the user. Using the history they are able to return more personalized results because they know approximately the field of interests of the user. This personalized search can be omitted through API of the search engine. This can be pity for the programs and automatic querying in case they are only focused on certain topic.

2. *"**PageRank** is a family of well known algorithms for assigning numerical weights to hyperlinked documents (or web pages or web sites) indexed by a search engine. PageRank uses link information to assign global importance scores to documents on the web. The PageRank process has been patented and is described in U.S. Pat. No. 6,285,999. The PageRank of a document is a measure of the link-based popularity of a document on the Web."* [4]

3. *"**TrustRank** is a link analysis technique related to PageRank. TrustRank is a method for separating reputable, good pages on the Web from web spam. TrustRank is based on the presumption that good documents on the Web seldom link to spam. TrustRank involves two steps, one of seed selection and another of score propagation. The TrustRank of a document is a measure of the likelihood that the document is a reputable (i.e., a non-spam) document."* [4]

Chapter 2

Data Mining

"The goal is to transform data into information, and information into insight."

– Carly Fiorni

Gathering of a huge amount of data evokes the need of their automatic processing. People want to know whether there is some hidden knowledge and how the message can be used. But how to mine this hidden knowledge from the data? That is the reason why data mining has been developed. There have been many algorithms developed to extract knowledge from the data set. The process of mining data involves these tree key parts: *pre-processing*, *data mining* and *validation*.

2.1 Pre-processing

Pre-processing of data is an important step of data mining. Raw data are usually incomplete (there are missing values), noisy (contain impossible values, for example age is *-10*), inconsistent (expected values are different, for example date of birth is *11 December 1999* and the age is *35*) or they are not suitable for classification algorithms (Algorithm accepts numbers only, but we are working with text).

There are different reasons why data are dirty. Incomplete data are often the result of different thinking processes at the time when the data were collected and the time when they were analysed. It is also the failure of software or hardware. This failure can be responsible for noisy data too. As a result of this, data contain some outlying or occasional values. Another producer of noisy data is of a human kind. When the results are collected manually, they may contain measure mistakes or when the person who is processing data is not focused on work he inducts mistakes too. The inconsistency of data is not so common but may occur too. It can be caused by

joining of data from different sources or by the software or human failure.

There have been developed several techniques how to handle wrong attributes or instances. Missing values are estimated by other data from data set or by external knowledge about the problem. Identified outlying values are removed or data are normalized. For inconsistent data there is required some external knowledge about problem to decide which values are correct, which values to remove and which values to leave or change. Duplicate records are considered as dirty data too and they also need cleaning. These techniques are only few from the large number of all the techniques used in pre-processing. The techniques depend on the nature of the problem and we have to pick the suitable one for our data.

2.2 Data mining

When our data is purified we can apply some techniques of data mining to extract previously unknown and potentially useful hidden knowledge. There have been developed a lot of algorithms for mining the knowledge. We can divide them into several groups according to what problem they can solve. The main groups are: *Classification*, *Clustering* and *Association rules*.

Classification is a process when the algorithm tries to categorize instances to predefined classes. The task of the algorithm is to find common features of learning instances which belongs to the same class. Classification is the most common problem therefore the number of algorithms solving this task is quite high. We can sort them to several categories: *Bayes*, *function*, *meta*, *rule*, *tree classifiers* but there is even more of them.

Clustering is similar to classification problem. The difference between clustering and classification is that for the clustering task we do not know categories. Algorithm itself tries to find similar attributes and compute their distance from other instances. Documents belong to one cluster if they have low distance from other documents belonging to the same class. There are two different types of algorithms: algorithms which require to know in advance the number of categories present in data and algorithms which guess the number of categories by themselves.

Association Rule mining algorithms are methods for discovering hidden relationships in data sets. They try to find rules applicable for instances in data set. For example they may find the rule: "*When somebody buy diaper he*

tends to buy beer too." in the sales data. Seller then may decide what promotion he may provide or how to sort goods in the store.

2.3 Validation

There are various ways to assess how the mined knowledge is able to cover new examples. Sometimes during the training phase the algorithm learns to recognize learned data, but when we provide new data it can get into troubles. This state is called *overfitting*. To see how accurate our model is we can use several techniques and compute various statistics.

Statistics

The computation of statistics for the model is an essential part of data mining. When we classify and divide examples into two classes as *positive* and *negative* we can compute *Accuracy*, *Precision* and *Recall* based on the *Confusion matrix* (Table 2.1).

		Prediction outcome	
		Positive	**Negative**
Actual value	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 2.1: Confusion matrix

Then statistics are defined as follows:

$$accuracy = \frac{\|TP\| + \|TN\|}{\|TP\| + \|FP\| + \|TN\| + \|FN\|} \tag{2.1}$$

$$precision = \frac{\|TP\|}{\|TP\| + \|FP\|} \tag{2.2}$$

$$recall = \frac{\|TP\|}{\|TP\| + \|FN\|} \tag{2.3}$$

Methods

Split data set

One of the statistical approach how to check the model, is to divide data into training set and test set. The training set is used during the phase of learning model. When the predictive model is created then the test phase follows. In this phase we submit test examples and compute statistics of model prediction. The most common question is how to split data set to training set and test set. It is also hard to answer what the best number is. The most common used split is around $\frac{2}{3}$ (60-80%) for the training set and $\frac{1}{3}$ (20-40%) for the test set.

Cross-validation

Another widely used technique is cross-validation. The test set is split to more subsets. In one round of testing one subset is used as a test set and other subsets are used for training. Every subset is used as a test set and the computing of the final statistics is based on the results of all the test rounds. There are two basic approaches of how many subsets to create:

- When we split the data set to K equal subsets we call it *K-fold cross-validation*. One subset is always left for testing and the rest $K-1$ subsets are used for learning. In this case the learning and evaluation is repeated K-times.
- *Leave-one-out cross-validation* is similar to *K-fold cross-validation* but in this case the K is equal to N, where N is the size of data set, which means there is always only one example used for validation and the rest of data are used for learning model. This model of validation is quite expensive and unsuitable for too big datasets.

Chapter 3

Web Person Search

"In the end, it's still best to wait for the one we want
rather than settle for what's available.
It's still best to wait for the one you love
rather than to settle for the one who's around.
It's still best to wait for the right person,
because life's too short to be wasting it on the wrong one."

– Leslie Sheh T.[34]

The amount of the information on the Internet is constantly growing. With the enlargement of the Internet the number of personal information posted in there grows too. The most current trend are social networks where people put a lot of personal information. People are sharing their opinions on various blogs, discussing on forums or just creating their own personal homepage. It is still more and more common for people to share documents with their work and people keep putting everything online. Companies use the web as a good market place or they use it as a presentation medium. All these small pieces of information create one huge source medium. Whether we want or not our name or nickname occurs on the Internet more and more often. It is still easier to track users, their work, field of interests and privacy too.

The reasons for searching people on the Internet are various. Businessmen can search for similar field of their business and for the people who are interested in the similar area. There is a competition between companies and the fight for talented people is very common. Companies are also looking for partnership and cooperation with other firms. Non-profit organizations or organizers of events often need to find sponsors. It is more likely that the person interested in the similar field of interest will support the organizations more than somebody who is not interested in the issue at all. But not only businessmen are looking for other people. There might be

also some personal reasons to search for people. Somebody needs to find contact information about somebody else. Or a person does not have actual information and needs to update it. Another reason arises when we know or meet somebody and we need to find out more information about that person. We might have our personal reasons, but nowadays it is also important if we care about our safety too. Especially when we know the person only via the Internet being cautious is important.

3.1 Who can we find?

Is really every single person available on the Internet? Can we find anybody? It is not hard to answer that question and the answer would probably be no; we cannot. We can split people to a few groups by their presence or absence on the internet.

The most common case is when people put information about themselves there. They usually use the social nature of the Internet – various blogs, forums, social networks, homepage and other ways to express themselves. Another group of persons who appear on the Internet are famous people and celebrities. They do not need to put information there, because other people make it for them. Also people working in firms are of a similar group. If they work in higher job positions or they own a company we can find at least contact information on their company page. We can find also active people joining or leading various activities. They usually want to present the activities they do and put information about them on the Internet. Another big group are people working in government. They are responsible for public affairs so they have their contact information available too. We cannot forget also the group of people joining various competitions or petitions. They put their contact information there and sometimes leave them publicly available.

On the other hand, there are also people who are difficult to find. The most spread and the largest is the group of people who do not use computer or the Internet. Also paranoid people or people protecting their own privacy do not share a lot of information about themselves. Of course this only applies in the case if nobody posts anything about them.

3.2 Problem definition

Definition 1 *Let q be a query for search engine with the name of the person we search for. Let $S = \{S_1, S_2, \ldots, S_m\}$ be a set of documents returned by search engine E. Let $PD = \{D_1, D_2, \ldots, D_n\}$ be a set of documents containing the name of searched person where $PD \subseteq S$ and $m \geq n$. We suppose every document D_i contains only information about one real person. Set PD contains k persons. Input for algorithm is query q and output is set of k clusters of documents D_i belonging to one real person.[39] [44]*

We post a query q to a search engine and we receive set of documents S. The query can be modified[1] by search engine and returned set of documents does not necessarily contains searched query. Query modification can be considered as a good but there are also situations where it causes more harm than profit. For example when we search for the name it is good when it finds also inflected words. On the other hand, when instead of *Mahrík* (person's surname) it finds *Máhrik*, the results may be misleading.

Even when all found documents contain only one name, there can be more physical persons with the same name. The algorithm for finding people on the web has as its input a set of documents with person's name. The task for the algorithm is to sort these documents according to real persons who occurred there. As an output of the algorithm there are groups of documents where each group represents one real person and the documents belonging to this person.

Named entities

When we want to identify person correctly, we need to know some facts about it. These facts are in the field of *information extraction* also called *named entities*. There is no exact definition of what the named entities are. On the *WebKnox Blog*[2] they tried to explain it by the means of "*concept*" as: "*A named entity is a collection of rigidly designated chunks of text that refer to exactly one*

1. Search engines use various techniques to modify original user query to provide more accurate search results. Query preprocessing actions can be found for example in patents: *US 2009/006365 A1*[21], *US 7472113*[41] or a good article on what actions search engine takes is available here: `http://www.infotoday.com/searcher/may01/liddy.htm`. It is just for overview because it is not a part of the aim of this work.
2. *WebKnox* is a question answer system. Service is available on the URL: `http://webknox.com/`

or multiple identical, real or abstract concept instances. These instances can have several aliases and one name can refer to different instances"[42].

The main task of Named Entity Recognition systems is to identify and extract named entities from a document. There are several categories of named entities such as: names of persons (first, last, other names), organizations (organization name, job positions); locations (address, URL, GPS coordinates); expressions of times (birth date, anniversary dates); phone numbers; e-mails; quantities; monetary values; percentages and others.

Jiang et al. identified eight important named entities when we search for a person: *Name*, *organization*, *address*, *profession*, *telephone number*, *date of birth*, *e-mail*, *URL*[16]. Not all of them are present on the Internet, but when they occur together it can be a strong identifier of a person. But not only named entities may help to identify person. Yoshida et al. found when *Compound Key Words* (CKW) occurs in a text it is a strong identifier of the person [44]. For example for the name *Bill Gates* the CKW may be "*chief software architect*" or for a baseball player it can be "*baseball stick*".

3.3 Problems connected with WePS

If we have already tried to search for some persons, maybe we have had some troubles. One name refers to more physical persons and sometimes it can be hard also for a man to decide who is the right person without any external knowledge. Some information can be also out of date and can lead to some mistakes. When we are trying to make this process automatic, we can have even more troubles. Documents are in different formats and encodings, so at first we need to transform them to common format with common encoding. Also when page contains more names and contact information belonging to those persons. It is hard for algorithm to decide which contact information belongs to which person. Another problem is with pages which use *JavaScript* to generate their content. User can see all information in browser, but algorithm can see only *JavaScript* code. It gets even more difficult when the whole site is written in *Flash*. We have to extract text out of the *Flash* program.

3.4 Related work on WePS

The problem to find information about a person on the Internet is more known than it might seem at first. From all incoming queries to web search engines 30% of them contain a person's name [3]. The first workshop on

Web Person Search was held in Prague in 2007. The need of this workshop arose from the growing number of people interested in this topic and also new web services offering finding people on the Internet. The main goal of this workshop was to build a testbed[3] corpus for development and evaluation of WePS systems. It was necessary to start an evaluation framework for WePS systems [3]. Another two workshops about this topic were held in 2009 and 2010 [4].

However, this topic is not interesting only as a challenge for researchers. Various commercial but also open source projects are trying to create such a system. The system should find information about people on the Internet, cluster them and try to sort found documents by physical people. Here is a short survey of such a systems:

GRAPE Web Person Search[16] is based on the $Carrot^{2}$[5] clustering engine. It is able to search through different web search engines API and collect results. You can choose from different clustering algorithms for clustering results or you can just write your own if either fails to comply with your requirements. *GRAPE* defines its own clustering algorithm focused on people clustering.

Commercial systems

Pipl is focused on systems not indexed by web search engines [29].
Pipl is available on URL: `http://pipl.com/`

iSearch searches in data previously aggregated from publicly available records [15].
You can find *iSearch* on URL: `http://www.isearch.com/`

123people searches through web pages not necessarily indexed by other search engines [2].
Service is available on URL: `http://www.123people.com`

Wink offers more options how to refine search query [25].
The system is available on URL: `http://wink.com`

3. "A platform on which an assortment of experimental tools and products may be deployed and allowed to interact in real-time. Successful tools and products may be identified and developed in an interactive, evolutionary, interdependent process."[10]
4. Web page of workshops: `http://nlp.uned.es/weps/`
5. $Carrot^{2}$ can be found on URL: `http://search.carrot2.org`

Yahoo! People search integrates people search to *Yahoo! web search*. Is the same like you use *Yahoo! search* with following controls: `first:"<name>",last:"<surname>",city:"<city>",state:"<country>"` [43].
You can find *Yahoo! people search* on the URL: `http://people.yahoo.com/`

PeekYou tries for every public weblink answer the question, "*Who made it?*" or "*Whom is it about?*" [28].
PeekYou is available on: `http://www.peekyou.com/`

intelius Daily update its database and accessing to various public records sources from their data partners [14].
Intelius service is available on the URL: `http://www.intelius.com`

3.5 Disambiguation model

The main task of the disambiguation algorithms is to split the set PD and make k clusters. Every cluster should represent one real person. This can be a difficult task, because the same person occurs in different contexts and usually with very little and often disturbing elements [11]. Now we describe some used approaches for disambiguation persons and documents associated with them.

TruthFinder [11]

Web pages contain a lot of controversial information about a lot of objects. Truth analysis intended to ascertain the correct information about each object. TruthFinder is using this two main heuristics to approve correct information:

- "*an assertion that several information providers agree on is usually more trustable than that only one provider suggests*"[11].
- "*an information provider is trustworthy if it provides many pieces of true information, and a piece of information is likely to be true if it is provided by many trustworthy web sites*"[11].

Graph making [39]

Is the approach where we construct a graph based on occurrence of named entities. Named entities represent vertexes and we can make edge between

vertexes when the named entities occur in the same document. The strength of the connection between named entities depends on weights of vertexes and edges. After we set weights we run clustering algorithm which will make clusters on that graph. Graph is split to sub-graphs where each sub-graph represents a certain person.

Fuzzy Ants [19]

This algorithm is inspired by nature. It is inspired by the technique of ants when they collect corpses to two heaps. In laboratory conditions ants always know where to put corpses. It is not specified how many heaps there should be. It is inspired by the idea that ants behaviour can be flexible and we can express it by condition `IF-THEN`. At the beginning of the clustering there is no initial partitioning. Ants can take one item or the whole heap of items. At any moment in time the ants can decide whether to pick-up or drop item or heap of items. The reasoning whether to pick-up item or heap of items is based on counting of documents similarity. Also dropping of items or heaps is based on counting of similarity between documents. By these simple steps ants make heaps which represent clusters and each cluster represents one real person.

Double stage clustering [44]

In this approach the words in a document are split to two groups: *strong* and *weak* words. *Strong* words can distinguish document between clusters and *weak* words cannot. *Strong* words are *named entities* and *compound keywords*. Other words in the document are treated as *weak* words. For example *Memory* and *algorithm* are *weak* words for *Programmer*. In the first stage of clustering algorithm distinguishes documents only by *strong* words. It makes clusters. Based on these clusters it sets weights for *weak* words. In the second stage of clustering algorithm also takes into account the weight of *weak* words and rebuild clusters.

Chapter 4

Business partner search

"If we are together nothing is impossible. If we are divided all will fail."

– Winston Churchill

When the reasons for finding people arises from our business the requirements for finding people may slightly differ from the original WePS task. When we search for our potential business partners we usually have a list of people. Out of this list we have to pick persons who are suitable candidates to reach out to offer a partnership. It means we have to look also for the scope of the work of that person. If this scope is similar to the one we are looking for, this person is more relevant for us. We also need to find some other information about the person or the person's company. When we are looking for sponsors, it is good to know if the person has already sponsored some event. If so, it is more likely for them to sponsor also another event.

4.1 Problems connected with searching for business partners

When we are searching for business partners we might be confronted with several problems. We can find a lot of information about person but this information might not be associated with the company the person works for. And if the company sponsored an event we might not find this out. The problem appears also when companies cooperate together. On the page of the company can be mentioned the name of our searched person. But this person does not necessarily have to work in this company and algorithms can be confused by this. Another kind of problem is that search engines do not index the whole internet. So some pages such as business registers are not available through this search.

4.2 Search by hand

Our first aim was to understand how to find people and acquire some experience. So at first we tried to find people on Google by hand and write some notes why have we chosen that person or why we discarded the found document. By our personal experience we proposed pseudo Algorithm 1.

When we are searching for people by hand and if we try to find concrete person we also use multiple queries with some refinements. Refinements are based on *Named Entities* or *Compound Key Words*. When we are searching for a doctor we also include words like *surgery*, *M.D.* or *nurse*. By query refinement we get more deeper to the specific area of interests of the concrete person.

4.3 Information resources

When we are searching for people we usually and mainly use search engines for searching. As we mentioned before, not the whole content of the Internet is indexed. There are some pages or whole servers such as business registers, job portals or other useful sites which are not indexed. Especially business registers are for us a rich source of information. Now we will make short review of useful information resources for this problem. We will primarily focus on the resources in the Czech and Slovak republic. Though similar pages can be found also in other countries.

4.3.1 Information about companies and business persons

This is the best source of contact information. We can find person's personal details there as well as other companies which this person owns or is a co-owner of. It is also possible to search for people whom the person may know in case they are co-founders of the company.

ARES

ARES Business register (in Czech Republic)

- "***A**ccess to **R**egisters of **E**conomic **S**ubjects / Entities is an information system allowing a retrieval of information on economic entities registered in the Czech Republic. This system intermediates a display of data from particular registers of the state administration (called source registers) in which the data concerned is kept.*" [24]

Algorithm 1 Pseudo code of searching by hand

Require: List P of possible business partners.
List PI of already identified persons.

1: **for all** $person$ in P **do**
2: Search $person$ in Business register
3: Search $person$ in Social Networks, Companies & Job Portals, Other
4: Search $person$ among friends of $personX$ where X is in PI or P
5: ***SearchOnWeb****(person)*
6: Try to cluster found contact information and persons
7: **if** Many found documents contain the same *word* possibly characterizing the same person **then**
8: Try to find another information or a different person by excluding *word* from query q
9: ***SearchOnWeb****(person)*
10: **end if**
11: **if** Documents contain *word* of type CKW (3.2) **then**
12: Use this *word* with next query q *//CKW represents company, areas of activity...*
13: ***SearchOnWeb****(person)*
14: **end if**
15: **end for**
16: **return** List PI of clusters each referring to particular person. There the items in each cluster are ordered by relevance.

1: **function** SearchOnWeb(Query q){
2: Search for $document$s in search engines
3: **exclude** $document$ **when**
4: Does not contain searched name
5: Searched name is split and includes words that are not names
//search: John Smith found: John XY .. YZ Smith
6: Searched title is not before or after name
7: **exclude end**
8: Extract main context of document
9: Extract contact information
10: Extract useful links and download them
11: }

- We can search for Economics subjects (persons) there and it will returns us XML document with found results about person.

- Address of the service: http://wwwinfo.mfcr.cz/ares/.

Friend Of A Friend – foaf.sk

- Project foaf.sk uses information from *Companies Register of SR*[1], *Tax Bureau of SR*[1], *Social Insurence Institution* and other databases. By graph algorithms they look for companies and persons and show deeper connections between them in Slovak Republic.

- We can easily explore who is co-owner of the company. It is easy to explore close persons of searched persons.

- Address of the service: http://www.foaf.sk

4.3.2 Information about staffing of companies

These pages write articles about moving people between companies. Which person goes to which position in another company. This can be useful when we are searching for managers.

- CIO BusinessWorld
  http://businessworld.cz/aktuality-a-zajimavosti/personalni-udalosti

- ITBIZ.cz – articles about who moved where between companies
  http://www.itbiz.cz/lide-ve-firmach-kveten-tri

4.3.3 Job portals

On these portals we can find the connection between names and companies. When companies publish ads they leave also the names of managers and some contact information about them on the portals. Here is a short review of job portals in the Czech Republic.

- jobdnes.cz, jobs.cz, sprace.cz, monster.cz, profesia.cz, spravnykrok.cz, prace.cz, abcprace.cz

1. Slovak Republic

4.3.4 Social networks

Nowadays social networks are the richest sources of information. People are sharing and putting there almost everything. But these information are not usually publicly available.

Facebook

- Very good source of information, but protected by user settings
- Pages are generated by *JavaScript*, we need run *JavaScript* on page code at first to get information
- Friends of searched person are very important – their friends could be also on the list of searched persons. By this we can better identify the person for whom we are searching for.
- URL: facebook.com

LinkedIn

- Portal oriented on job opportunities
- User profiles are oriented on Curriculum Vitae

Other social networks

ResearchGate , SciSpace, ScienceStage, Epernicus

4.3.5 Other sources

Web pages of companies and personal pages

Nowadays most of the companies have their own web page as a form of a presentation or also used for selling their goods. Individuals want to present their work or themselves as well so they make their own pages with contact information.

Chapter 5

Web documents pre-processing

"I tried a dozen different modifications that were rejected.
But they all served as a path to the final design."

– Mikhail Kalashnikov

Pre-processing plays an important role in the field of machine learning. Data usually contains a lot of noise, misleading, missing values or it is just not suitable for algorithms. This noisy data leads to more difficult training phase or the data makes the training phase impossible to happen. There is also another advantage of transformation or pre-processing of the data. It lets us look on data in a different way. We can see some other relationships in data and we can try to use other methods of machine learning. There is a close connection between pre-processing and visual analysis, the use of which can be very helpful [17]. We can imagine the connections between different variables and their values better.
There are two types of pre-processing:

- **Supervised** Learn data are already known.We can make transformations and analyse data better before applying machine learning or data mining algorithms. For example patient data about cancer or classifying newspaper articles by their topic. We can also use the visual analysis to understand problem better.

- **Unsupervised** We collect data directly before pre-processing from the stream or another source. We apply only general pre-processing methods suitable for specific type of data. For example the analysis of a direct audio call.

There is a lot of possibilities how to create the document. When we download documents from the Internet, we have to take into account most of the possibilities. At least the most used.

5.1 Encoding

Documents are stored in different encoding depends on the characters used and the user's possibilities. The most common encodings are listed in Table A.3. When we want to process documents with different encodings, it is better to unify encoding of all downloaded documents. When we convert documents to common format, we can process them more easily. As we know, encoding carries different characters so we have to convert documents to encoding which will cover used characters. UTF-8 is one of encoding which can wrap all characters of Unicode standard [37], so it is sufficient for all characters, symbols and scripts of major languages of the world [36].

5.2 Format of documents

There is a huge variety of documents on the Internet. People create documents in various programs and share them in different formats. Luckily, there are some documents more widespread than others. The most common formats in which people usually publish text are indexed by Google. The list of these files is in Table A. For our task are first six types the most important: *Web Pages*, *PS/PDF documents*, *MS office documents*, *Open/Libre office documents*, *RTF documents* and *Flash animation*. These types may contain important information about the person we search for.

Web pages

The main portion of information on the Internet is contained in the *HTML* pages. Extraction of the text from this format is quite simple when pages are valid. It is enough to remove *HTML tags* from the document. When pages are not valid and contain errors we should validate them, because *HTML* parsers can display an error. Good valid code maker is *HtmlCleaner* [38] written in *Java*. It reorders elements of the web page and returns valid *XML* code. The usage is quite simple, but when we run it on large *HTML* page (hundreds of kilobytes) it may require more memory.

Some web pages uses *JavaScript* to display the content. When we download simple HTML code, some information can be missing. Before text extraction we should run *JavaScript* on the page. Modified page should be made valid and then the text content can be extracted. To run *JavaScript* on the page we can use *HtmlUnit* library [12]. *HtmlUnit* is a *Java* web browser without GUI. It downloads the page and runs initial *JavaScript*.

Table 5.1: Apache POI [18] format support

Program	Extensions
Microsoft Office Excel	*.xls, *.xlsx (97-2007, Word 6, Word 95)
Microsoft Office Outlook	*.msg
Microsoft Office PowerPoint	*.ppt, *.pptx (97-2007)
Microsoft Office Publisher	*.pub
Microsoft Office Visio	*.vsd
Microsoft Office Word	*.doc, *.docx (97-2007)

PS/PDF documents

PostScript or *Portable Document Format* are becoming more spread and people use them as suitable exchange format. It is because of their platform and their printer independence. Unfortunately this format does not contain pure text and we have to extract it. The extraction process may not be absolute especially when the document contains special characters or diacritic. As a good extraction library we can use *Apache PDFBox* [9]. Library can handle also special and diacritic characters. Although this library can extract text from the PDF, there are also documents made by scanning. These documents contain text, but the text is a part of the image. In this case we have to use some *Optical character recognition* (OCR) software to extract text from such document.

Microsoft Office documents

Microsoft Office is the most used software for creating office documents, spreadsheets, presentations and other documents. People also publish and exchange information created by this suite. Because of *Microsoft Office* using their own proprietary format we need a tool to extract the text from documents.

Apache POI [18] is a library which is able to extract text from the file types listed in Table 5.1. The usage of this library is quite simple. It is also able to extract embedded objects. That means if there is a spreadsheet inside of *PowerPoint* presentation as embedded object, it is able to extract text from this object too.

Open (Libre) Office documents

Open Office is another suite for creating office documents. Unlike *Microsoft Office* it uses *OpenDocument* (ISO/IEC 26300:2006) standard [8] for storing file content[26]. This standard is based on XML so extraction of the text from the document is quite easy and does not require special libraries. We need to unzip the file and open `content.xml` file. Then we just need extract the textual content of XML file.

RTF documents

Rich Text Format was introduced by Microsoft company in the latest version 1.9.1 [23] as inter-platform exchange format. Default charset was *ASCII* but from *Word97* it supports *Unicode* too [6]. Because RTF can internally contain only 8-bit values, it uses control words to encode 16-bit[1] *Unicode* characters [6] therefore we have to convert them.

Flash animation

Some companies do not use simple HTML page as their web presentation. They want special effects and design to be more compelling. Therefore they choose flash technology as their presentation. The main disadvantage from the point of view of text extraction is that it is a program and we cannot simply get its content. We need flash decompiler which is able to extract this textual content. *Swfmill* [35] is a flash compiler and decompiler which is able to convert *SWF* files to *XML* and vice versa [35]. When we have *XML* file we can easily extract textual content. It is stored in attribute `initialText` of `DefineEditText` tag.

5.3 Diacritics

In most of the languages it is common to put some special accent for letters in the word. Diacritics is mainly used to change the sound value of the letter to know how to pronounce the word. In these languages it is better to have a text with diacritic marks, because when we remove diacritics some words can be merged into one word. Unfortunately, some web pages like blogs or social networks contain text without diacritics. In that case, it is better to put diacritic back or when it is not possible, remove diacritic from

1. RTF 1.9.1 specify only 16-bit *Unicode* encoding. But only *Unicode* in version 1.0 (year 1991-1995) was 16-bit. *Unicode 2.0* standard encodes characters in a 21-bit code space. [37]

both documents to be able to compare them. In our case when we are interested mainly in the Slovak or Czech language there exist tools for adding diacritics into texts.

The Czech language has the program called *CZACCENT* [22]. The program was developed at Masayrk University in Brno by Pavel Rychlý and it is able to restore diacritic in 95,65% cases. As for the Slovak language there is one program *diacriticconverter* developed by Matej Sabo from the Slovak University of Technology in Bratislava [31]. The accuracy of this program is 83,54% for general text including names, surnames, geographic objects and so on. Success of the program without these words is more satisfactory and it is around 97,51% [31].

In case when we need to compare some Named Entities, it is better to remove diacritic. It is less common that Named Entities after removing diacritic signs are merged into one word and we have a better chance to match them. The list of supported diacritics symbols and letters which can be removed is showed in Table A. The list is not complete, because Java cannot handle some diacritic signs and splits them as two different characters.

5.4 Stop List

When we analyse textual documents they might contain also words that are not significant for the analysis. The words occur in the text but do not have any predictive value from the point of view of the problem. In this case it is better to identify and remove these words before analysing the text. The list of these words is strongly dependent on the problem and also on the data set we have. If the words are not removed the algorithm might be confused by them and does not provide satisfactory results.

5.5 Bag of words

Because most of the algorithms are not able to work with text, we have to convert it into some other form. The technique of conversion of the string to the list of words is called as *Bag of words*. We split text by some rules (by words) and then we can apply other techniques of pre-processing to make data processable.

5.6 Stemming

Words in a text occur in various forms. We can find words in plural, past tense and other forms. The technique used to find the base – the stem of the word is called *Stemming*. It helps us unify words with the same meaning but different spelling. Another similar technique used to unify also different verb inflections or noun declensions is called *Lemmatization*. It helps us to group together even more words and analyse them as a single item.

Chapter 6

Classification

"The classification of facts, the recognition of their sequence and relative significance is the function of science, and the habit of forming a judgement upon these facts unbiased by personal feeling is characteristic of what may be termed the scientific frame of mind."

– Karl Pearson [27]

When we try to find business partners there are various ways how to handle the problem. Because the problem is similar to *Web Person Search*, the first option of how to solve the issue is to cluster downloaded pages by persons. By clustering we put all the documents belonging to real persons to one cluster. However, is it really necessary to find all persons with given name? The above mentioned approach is good when we search for people with the same name and we do not care about the person's field of interests.

When we search for business partners we look for people working in the similar area of work. This observation can be helpful for initial filtering of documents because a lot of namesakes[1] do not work in the same area as the one where the searched person works. Therefore at the beginning we can filter documents by the area of interest. It may happen we also filter out some of documents with the person we are looking for. This is not a big problem, because when we want to find documents about a specific person we do not find them by first query and we have to use multiple queries. For example, when we search for people by hand, we also exclude or add some words as query refinement.

6.1 Collection of data

As the first step in gathering data we had to find influential computer scientist on the Internet as well as the pages where the name of the data occurs

1. A person having the same name as another [7].

or where the data is referred to. We used the data from the page of *Czech Society for a System Integration*[2]. This site contains the list of the companies[3] focused on information technologies. We went through the whole list and we checked every site whether there is a list of employees available. We gathered all employees working on a higher position in the company but only in the case they are computer scientist. The list of gathered people is available in the Attachment A.4. Out of this list we picked mostly graduated people but also persons without college education. These selected people are shown in Table 6.1. For all the people we gathered publicly available documents from the web.

Company	*Person's name*
2N Telekomunikace a. s.	Oldřich Stejskal
ABRA Software, a. s.	David Rožek, Ing. Martin Bok, Ing. Martin Jirmann, Ing. Petr Nejedlík, Ing. Petr Vacek
AG COM, a. s.	Ing. Viktor Horák, Mgr. Milan Janda
EG - Expert, s. r. o.	Ing. Hana Štiková, Ing. Josef Chrástek, Ing. Petr Paška, Ing. Petr Šnyta, Ing. Stanislav Koukol, Ing. Stanislav Kuřík, Mgr. Lidmila Vránová
IPEX a. s.	Ing. Pavel Píštěk, Ing. Petr Chruňák
Kapsch, s. r. o.	Ing. Karel Feix, Ing. Zbyněk Juřena

Table 6.1: List of searched people

Gathering data

For given list of selected people we search for information on the web by two major search engines – *Google* and *Bing*. *Yahoo!* and *Bing* return simi-

2. ČSSI (Česká společnost pro systémovou Integraci) is a non-profit organization for exchanging information and views in the area of intormation systems, Information and communication technologies. Associate users and suppliers of services and products of information and communication technologies. URL of the page: `http://www.cssi.cz/cssi/` [5]

3. Directory of members is available at `http://www.cssi.cz/cssi/adresar-clenu`

lar results, the only difference between them is in sponsored links, so we can omit *Yahoo!* search engine[4]. We used publicly available API of search engines to collect data. For each person we downloaded the top 50 results of each search engine. Sometimes search engines returned us equal links so we joined them and downloaded only one document.

Documents preprocessing

The variety of formats on the Internet is huge. When we download a document from Internet, we have to identify the format of the document. The format is identified by *mime-type* returned from the server. For textual documents it is also important to know encoding. Documents are published in different encodings and it is better to convert them to common one. We used *UTF-8* as encoding for all documents because it covers most of the languages and scripts there are. From non-textual documents like *PDF/PS documents*, *Microsoft Office documents*, *Open/Libre Office documents*, *RTF documents* or *Flash* we extracted text by tools discussed in the Chapter 5.2.

When we extracted the text we also applied other filters on textual documents. Because a lot of documents were web pages, we converted HTML entities[5] for their representing signs and removed HTML tags[6]. We converted also all characters to lower case. Moreover, special characters in the text such as `[~/,+:()...-©#[]{}\"*``";%.!]?<>®` do not have significant impact on this type of classification task so we removed them as well.

When search engine returns us results it usually does not care about the language of the documents. Especially when we search for a person's name, the namesake might live also somewhere else in the world, so the language of the document might be different as well. However, if we want to find common features of a group of people, we cannot do it with documents written in a different language. We identified the language of all documents with the *language-detection* tool and for classification we selected only documents written in the Czech language.

---

4. *Yahoo!* has paid usage of their API
5. Reserved characters in HTML document are replaced with character entities, for example: less sign < can be represented as &*lt*; or &#60 therefore we converted it to <. [40]
6. From web pages were removed HTML tags but we preserved attribute values. For example the tag `<img src="img.png" title="Title of the image." >` was converted to the text "*img.png Title of the image.*"

Removing stop words

Since in the text there might also occur words which do not help us to distinguish between classes, we can remove them as well. The following types of words were removed from documents used for classification :

- Education titles (*ing*, *mgr*, *paeddr*, *drsc*, *dis*)
- Names of companies where searched persons worked (*2n*, *telekomunikace*, *abra*, *ag*, *eg*, *ipex*, *kapsch*)
- Surnames of searched persons (*feix*, *stejskal*, *rožek*, *bok*, *nejedlík*, *vacek*, *horák*, *janda*, *paška*, *štiková*, *chrástek*, *kuřík*, *koukol*, *šnyta*, *vránová*, *chruňák*, *píštěk*, *juřena*, *jirmann*)
- Other person's surnames occurred often in data (*blažek*, *čech*, *čermák*, *černý*, *doležal*, *dvořáková*, *fiala*, *hynek*, *jelínek*, *krejčí*, *kučerová*, *moravec*, *müller*, *musil*, *navrátil*, *němcová*, *nováková*, *novotná*, *pokorný*, *rené*, *pospíšil*, *procházková*, *růžička*, *sedláček*, *šimek*, *svobodová*, *vaněk*, *veselý*)
- Names of cities (*meziříčí*, *teplice*, *karlovy*, *brod*)
- List of birth names[7]

Such modified data were used for classification – as a first step classification by hand and after that we learned classifier to distinguish between these classes. This set of documents used for classification we shall call the **main learning set**.

6.2 Classification of pages

Our first task before learning classifier was to create learning set. We downloaded and preprocessed documents about influential computer scientist from the Internet. Although not all the downloaded documents were exactly about person we were searching for. We had to sort the documents and divide them into categories. We decided to divide them to the four following categories: *MANAGER*, *INFORMATICS*, *OTHER* and *INFO_MANAGER*.

7. List contained 4890 names and it is available at: `http://krestni-jmena.cz/seznam/?perpage=2500&orderby=name&filts=0&filtc=0&filta=0&fullt`

INFO_MANAGER This group contains documents about the people we were searching for. These are the people working in the field of informatics providing that they work in higher position in the company. Also when the namesakes were computer scientist and they have higher position in the company they were included too.

INFORMATICS Is a group of documents talking about informatics. When the person mentioned in the document worked in the field of informatics we put such document to this group. It was not necessary for the person to work in higher job position. This group does not contain people working in the field of informatics in higher position as managers or company owners.

MANAGER In this group there are presented documents about managers. The document does not need to deal with a manager working in the field of informatics, nor is it acceptable. So this group contains documents about managers from all other areas of work.

OTHER This is the largest group of classified documents. There are all the documents which do not belong to any of the previously mentioned categories.

In spite of the fact that we are searching for influential computer scientist only, we divide the documents into four classes, even though it was not necessary. The reason is to perform also other kind of tests. We wonder whether it is possible to classify documents in other ways.

6.3 Training classifier

For the training of classifiers we used the **main learning set** as we described above. The final distribution of particular classes is shown in Figure 6.1. The real distribution of these classes over the web is not known yet, since there was no work posted about similar topic nor any research about the distribution of these classes was made. As we can observe there is a huge number of documents in the category *OTHER*. The baseline for the main learning set is around 86%. Seeing that baseline is relatively high we performed two different types of tests. The first one was when we just split the *main learning set* into two sets: training set and test set. The split was in 80:20 ratio: 80% of learning set and 20% of test set. In the second case the main learning set was split in the same way. The difference was we decreased the baseline of the training set to 60% by random removing examples of the class

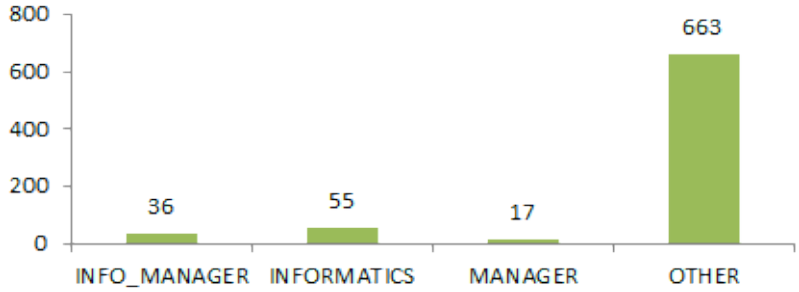


Figure 6.1: Distribution of particular classes

OTHER. However, just simple removing some examples from the *OTHER* class can lead to untrustworthy results. Therefore we ran classification with more learning sets. The baseline was decreased only for the training set. The test set stayed untouched with the original distribution of classes. The same applies for splitting the *main learning set* with 80:20 ratio. Therefore we performed more tests in both cases.

Chapter 7

Results

"Everyone ought to bear patiently the results of his own conduct."

– William Shakespeare

As we mentioned in the previous chapter, we performed more tests with more approaches applied. We executed each test with 56 different data sets derived from the *main data set*. To estimate how accurate the model is we used the test set with 20% of examples. Since it is important for this task not to exclude retrieved documents out of *INFO_MANAGER* class when they belong there. We used recall as a constitutive statistic. Especially the *recall* of the class *INFO_MANAGER*. The tables in this chapter contain only recall values and other statistics computed from the recall value. Algorithms in the tables are sorted by *INFO_MANAGER* recall as well. Other statistics like accuracy and precision are available in the attachment tables.

7.1 INFO_MANAGER classification

Our main aim was the classification of documents belonging to the group *INFO_MANAGER* in comparison to all other classes. The documents from the class *INFO_MANAGER* were used as positive examples in the learning set. On the other hand, the set of negative examples were the combination of the classes *INFORMATICS*, *INFO_MANAGER* and *OTHER*. As we mentioned above, we performed two kinds of tests. In both cases we split the *main learning set* with 80:20 ratio. The first kind of tests was performed with unmodified training set with original number of train examples of both classes. In the following text we will label it **full training set**. In the second case we removed train examples of *OTHER* class from the learning set to decrease too high baseline. This training set will be referred in the further text as **modified training set**. The test set stayed untouched in both cases with preserved distribution of classes.

Full training set

Among 25 tested classifiers the best results in the mean were achieved by *SGD*[1] and *SMO*[2] classifiers as we can see in the result Table 7.1[3]. Similarly the value of the recall for the *INFO_MANAGER* class was the highest achieved by these two algorithms. Despite of the fact that these two algorithms had the same result in the mean, *SMO* had slightly better result for the dispersion of *INFO_MANAGER* class. As the best tree-based algorithm was *J48* and results are comparable with the first two algorithms. Apart from that, from the results we can see that tree (wave underline) and function (line underline) -based algorithms distinguish data better than other types of algorithms do.

		Recall				*IM Recall*			
	Algorithm	**Mean**	σ	**Min**	**Max**	**Mean**	σ	**Min**	**Max**
1.	SMO	0.98	0.01	0.95	0.99	0.71	0.15	0.25	1
2.	SGD	0.98	0.01	0.96	1	0.71	0.17	0.25	1
3.	J48	0.96	0.01	0.93	0.99	0.69	0.18	0.25	1
4.	JRip	0.96	0.02	0.91	1	0.67	0.19	0.13	1
5.	Random Tree	0.96	0.01	0.93	0.99	0.67	0.17	0.25	1
6.	Logistic	0.97	0.01	0.94	0.99	0.67	0.16	0.38	1
7.	MultiClass Classifier	0.97	0.01	0.94	0.99	0.67	0.16	0.38	1

Table 7.1: Manager working in the field of informatics; original baseline

Modified training set

When we sort the results by *INFO_MANAGER* recall for the modified set we get order of algorithms like in Table 7.2[4]. In the first three places there are algorithms which achieved the best results also for the *full training set*. When we compare recall for the class *INFO_MANAGER* with previous results, we can see it is unexpectedly higher. In spite of the fact most of the algorithms in the weighted recall did not reach the baseline[5], the classifica-

1. *Stochastic gradient descent* is an optimization method for reducing an objective function written as a sum of differentiable functions.
2. Sequential Minimal Optimization. Platt's implementation of Support Vector Machine
3. All results are available in the attachment Table A.6
4. Full results are available in the attachment Table A.7
5. Baseline for the test set was 94

tion of documents belonging to the *INFO_MANAGER* class was better. Also dispersion of the *INFO_MANAGER* recall was slightly lower in comparison to previous results but higher for weighted recall. Likewise in the previous examples types of algorithm for this task are from the function, tree -based algorithms but also two meta classifiers.

		Recall				*IM Recall*			
	Algorithm	**Mean**	σ	**Min**	**Max**	**Mean**	σ	**Min**	**Max**
1.	SGD	0.95	0.02	0.89	0.99	0.83	0.14	0.5	1
2.	J48	0.87	0.05	0.74	0.95	0.83	0.13	0.38	1
3.	JRip	0.87	0.08	0.56	0.97	0.83	0.16	0.38	1
4.	MultiClass Classifier	0.9	0.05	0.75	0.98	0.81	0.15	0.5	1
5.	Logistic	0.9	0.05	0.75	0.98	0.81	0.15	0.5	1
6.	Simple Logistic	0.92	0.03	0.85	0.98	0.8	0.17	0.38	1
7.	Filtered Classifier	0.88	0.05	0.75	0.95	0.8	0.14	0.38	1

Table 7.2: Manager working in the field of informatics; decreased baseline to 60

Conclusion

We found out that decreasing of the baseline for the training set has an impact on the result of classification of documents to the *INFO_MANAGER* class. Algorithms are able to cover this class better when *INFO_MANAGER* documents are not lost between *OTHER* documents. This is important finding because we are mainly interested in pre-filtering retrieved documents from the Internet. We want to select only documents which belong to the *INFO_MANAGER* class. We do not need to detect whether the document belongs to *OTHER* class or not.

7.2 Two Stage Classification

We usually learn classifier on one learning set with positive and negative examples in one round. If we look at the class *INFO_MANAGER* from a different point of view we can see it as two kinds of documents: *INFORMATICS* and *MANAGERS*. The important question is whether it is possible

to learn two classifiers and test them in two stages. To prove this theory we used all four classes from the *main learning set* and created two training sets and one test set.

The first training set contained as positive examples documents from the class *INFORMATICS* and as negative examples documents from the class *OTHER*.

The second training set was composed of documents of *MANAGER* class. Randomly picked documents from the class *OTHER* were also used as negative examples.

Test set Finally, test set was created from the set of documents we are interested in: *INFO_MANAGER*. They were used as positive examples while negative examples were random documents of the class *OTHER*.

Training phase

During the training phase we learned two classifiers: one for the recognition of documents of the class *INFORMATICS* and the other one for *MANAGER*. Trained classifiers were connected in a row. We submitted a test document for the first classifier. The trained classifier then determined whether the document belongs to its class or not. When it does not recognized document, we considered the document to belong to the class *OTHER*. In the case the second classifier did not recognize the document, we considered the document to belong to the class *OTHER*. When no classifier recognized the document as the one of the class *OTHER*, we considered the document to belong to the class *INFO_MANAGER*, since it belonged to both of the classes *INFORMATICS* and *MANAGER* at the same time. The schema of the evaluation phase is shown at Figure 7.1. Again we created two different training sets. The first one was unmodified with preserved distribution of the classes. The second training set was modified and random documents of the *OTHER* class were removed to decrease baseline to 60. The training set stayed untouched again with preserved distribution of classes. The baseline of the test set was 84 in this case.

Informatics-Manager

In this two stage classification process the documents were classified for *INFORMATICS* class at first. In the case they belong to this group they were

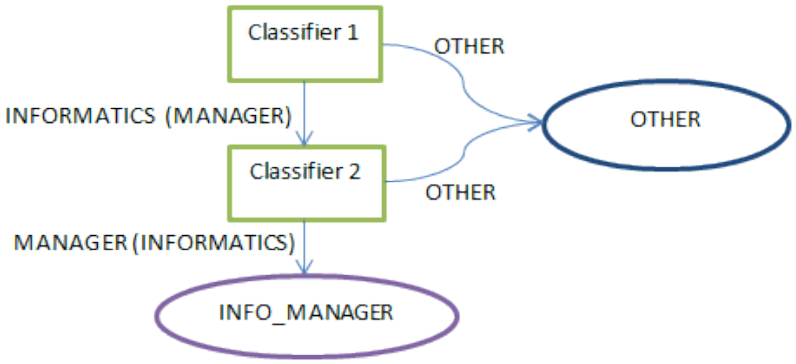


Figure 7.1: Evaluation phase for two stage classification

classified by the second classifier whether the documents belong to *MANAGER* class. Even the training set was different not only in the way of examples but also the results of classification were good.

Full training set

As we can see in Table 7.3[6] the classification of new examples of the *INFO_MANAGER* class when we trained two different classifiers on *INFORMATICS* and *MANAGER* class are good. They are comparable and in some aspects they over-

		Recall				*IM Recall*			
	Algorithm	**Mean**	**σ**	**Min**	**Max**	**Mean**	**σ**	**Min**	**Max**
1.	NaiveBayes	0.83	0.02	0.79	0.87	0.75	0.01	0.72	0.75
2.	SMO	0.95	0.01	0.93	0.97	0.73	0.03	0.67	0.83
3.	Simple Logistic	0.93	0.02	0.89	0.96	0.71	0.08	0.5	0.83
4.	LogitBoost	0.93	0.02	0.87	0.95	0.69	0.07	0.47	0.81
5.	MultiClass Classifier	0.93	0.02	0.87	0.96	0.68	0.12	0.33	0.83
6.	AdaBoost M1	0.92	0.02	0.87	0.95	0.65	0.08	0.31	0.75
7.	PART	0.9	0.02	0.85	0.94	0.64	0.11	0.39	0.83

Table 7.3: Two stage clustering – Informatics-Manager; Original baseline

6. Full results are available in the attachment Table A.8

reach the *original classification* of *INFO_MANAGER* and *OTHER* classes. The best result in *original classification* had weighted recall *0.98* and recall for the class *INFO_MANAGER* was *0.71* by *SMO* classifier. These results are not good only for the recall but also the dispersion of *INFO_MANAGER* recall is significantly lower as in the *original classification* when it was *0.17*. It is also worth to mention that the baseline in the *original classification* was around 95 and in this case it was around 86.

Modified training set

When we modified the training set and left the baseline on the value 60 for both classifiers the results became even better likewise it was in the *original classification*. As we can see in Table 7.4[7] *SMO* had again the best results with the highest *INFO_MANAGER* recall as well as the weighted recall. When we compare these results with *original classification* the best algorithm (*SGD*) had weighted recall *0.95* with dispersion *0.02* and *INFO_MANAGER* recall *0.83* with dispersion *0.14* we can see that these results are quite similar.

		Recall				***IM Recall***			
	Algorithm	**Mean**	σ	**Min**	**Max**	**Mean**	σ	**Min**	**Max**
1.	SMO	0.92	0.02	0.86	0.95	0.82	0.03	0.75	0.86
2.	Simple Logistic	0.88	0.03	0.76	0.93	0.8	0.06	0.58	0.92
3.	J48	0.82	0.04	0.73	0.88	0.78	0.1	0.42	0.94
4.	Logistic	0.89	0.04	0.8	0.95	0.77	0.09	0.53	0.89
5.	MultiClass Classifier	0.9	0.02	0.85	0.95	0.77	0.1	0.44	0.86
6.	LogitBoost	0.85	0.03	0.77	0.9	0.76	0.09	0.5	0.89
7.	Classification Via Regression	0.82	0.04	0.75	0.91	0.76	0.09	0.56	0.92

Table 7.4: Two stage clustering – Informatics-Manager; decreased baseline to 60

Manager-Informatics

For this kind of classification the document was at first classified whether it belongs to *MANAGER* class and consequently we decided whether the

7. Full results are available in the attachment Table A.9

document also belongs to the *INFORMATICS* class.

Full training set

The results for unmodified training set are available in Table 7.5[8]. The first algorithm in the table overreached the results of *Informatics-Manager* classification in both weighted recall and *INFO_MANAGER* recall. Other algorithms had surprisingly low *INFO_MANAGER* recall even the same test set in one round of testing was used for all algorithms. Weighted recall was in average *0.86* over all algorithms.

		Recall				*IM Recall*			
	Algorithm	**Mean**	**σ**	**Min**	**Max**	**Mean**	**σ**	**Min**	**Max**
1.	NaiveBayes	0.93	0.02	0.88	0.97	0.84	0.06	0.69	0.91
2.	J48	0.85	0.03	0.77	0.92	0.45	0.16	0.14	0.86
3.	JRip	0.84	0.04	0.76	0.93	0.4	0.18	0.06	0.72
4.	Random Tree	0.83	0.03	0.75	0.93	0.38	0.15	0.14	0.72
5.	Simple Logistic	0.88	0.02	0.81	0.91	0.38	0.14	0.17	0.69
6.	PART	0.84	0.03	0.78	0.92	0.38	0.14	0.14	0.69
7.	LogitBoost	0.85	0.02	0.8	0.9	0.34	0.13	0.14	0.67

Table 7.5: Two stage clustering – Manager-Informatics; Original baseline

Modified training set

The modified training set again finished with better results as the one with unmodified training set. The results are available in Table 7.6[9]. But similarly the recall of *INFO_MANAGER* class decrease relatively fast. The best algorithm was again *Naive Bayes* as in the previous case.

Conclusion

Seeing the results of this two stage classification we can proclaim that this method is competitive to the *original classification* when we classified only on classes *INFO_MANAGER* and *OTHER*. When we trained classifier on

8. Full results are available in the attachment Table A.10
9. Full results are available in the attachment Table A.11

	Recall				*IM Recall*			
Algorithm	**Mean**	σ	**Min**	**Max**	**Mean**	σ	**Min**	**Max**
1. NaiveBayes	0.85	0.06	0.71	0.93	0.9	0.06	0.71	0.97
2. Logistic	0.87	0.04	0.75	0.93	0.81	0.2	0.31	1
3. MultiClass Classifier	0.86	0.05	0.75	0.95	0.78	0.19	0.28	1
4. Classification Via Regression	0.74	0.07	0.5	0.84	0.65	0.16	0.28	0.97
5. SMO	0.87	0.03	0.78	0.94	0.65	0.16	0.36	0.89
6. Simple Logistic	0.8	0.04	0.7	0.87	0.6	0.14	0.28	0.83
7. J48	0.74	0.06	0.58	0.88	0.6	0.14	0.22	0.86

Table 7.6: Two stage clustering – Manager-Informatics; Baseline decreased to 60

the *INFORMATICS* and *MANAGER* classes, they were able to decide together, with relatively high recall, whether the document belongs to the class *INFO_MANAGER* with relatively high recall. The best algorithm for this task was *SMO* classifier. Following the results it is better to classify document whether it belongs to *INFORMATICS* class and then whether it belongs to *MANAGER* class.

7.3 Cost-Matrix

In the classification task we have to classify examples to one or more of the predefined categories. During the classification, most of algorithms consider all groups to have the same weight. However, the classification groups might not be equal and we might even reach better results by changing their weight. What will happen if we change the weight of *INFO_MANAGER* or *OTHER* class?

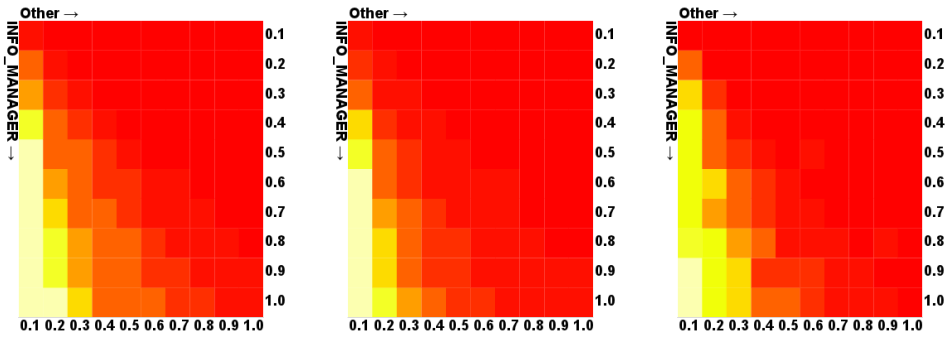
To search for a cost-matrix and to find good values for classes is not an easy task. There is no standard procedure of how to find the values. The cost-matrix is a square matrix where the size is equal to the number of the classes with zeroes at the diagonal. We are searching for all other values of classes. The values of the matrix are real numbers from the interval $< 0, 1 >$. The value *0* represents minimal impact of the group and the value *1* represents its maximal impact.

The range of values in the interval is infinite so we were searching only for a few numbers of interval with uniform distribution. We used numbers from 0.1 to 1 (both of them included) with a step *0.1* for each class. In the *weka* environment there is a classification algorithm *CostSensitiveClassifier* which for a given classification algorithm uses also cost-matrix and its weights during the classification.

Because the calculation of cost-matrix is computationally-intensive problem we did not count it for all algorithms again. We selected only the best algorithms and we computed cost-matrix for them. *SMO*, *RandomForest* and *NaiveBayes* did not have any valuable result therefore we present results only for *J48*, *JRip* and *SGD*. For all the following figures we can apply the rule: more red colour means better result of classification and more yellow (or white) colour means worse result.

Full training set

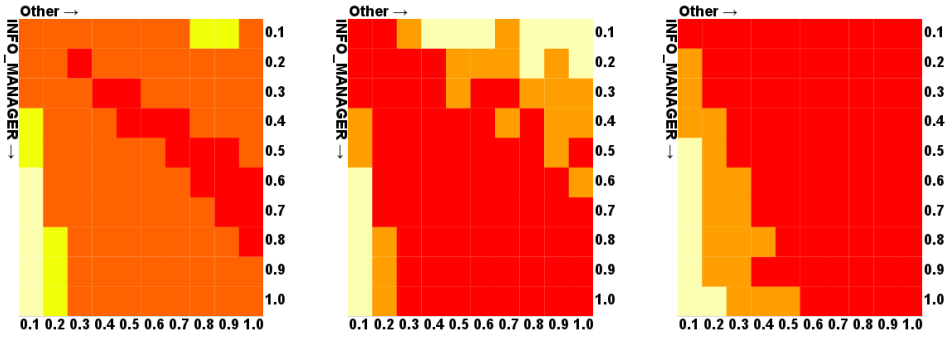
In Figure 7.2 we can see the results of *INFO_MANAGER* recall cost-matrix values. The red colour is slightly shifted for the class *OTHER* to number 1 and for *INFO_MANAGER* class to lower value for all three algorithms. It means that classification algorithm tends to classify more documents of the *OTHER* class as if they belong to *INFO_MANAGER* class. It can be caused also by big group of OTHER documents. On the other hand, when we look at the visualization of cost-matrix values for weighted recall in Figure 7.3 it looks differently. For *SGD* algorithm there is not such a big dif-



(a) J48 (b) JRip (c) SGD

Figure 7.2: Visual analysis of cost-matrix results of IM recall for unmodified training set

ference, but two tree algorithms have more balanced ratio of *OTHER* and *INFO_MANAGER* weight values. When we look at the values of classification, not only in the graph, we can see also some improvement. *SGO* algorithm has only slight improvement unlike the one of *INFO_MANAGER* recall for tree based algorithms. The results of *J48* classification are better by 7% and *JRip* has improved by 11%.



(a) J48 (b) JRip (c) SGD

Figure 7.3: Visual analysis of cost-matrix results of recall for unmodified training set

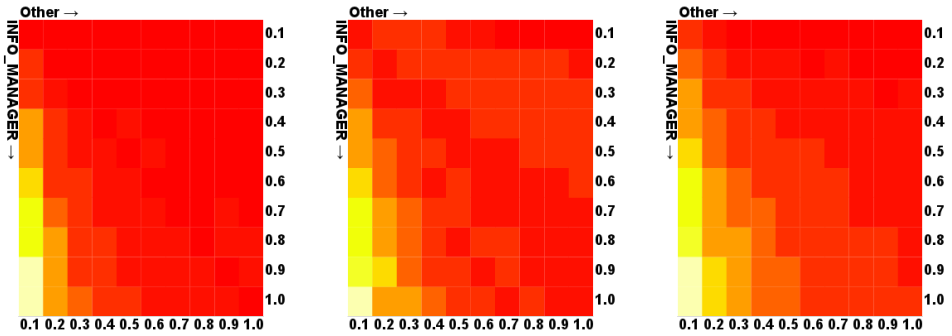
	Cost-Matrix		*Original*	
	Recall	**IM Recall**	**Recall**	**IM Recall**
SGD	0.98	0.73	0.98	0.71
J48	0.97	0.76	0.96	0.69
JRip	0.96	0.78	0.96	0.67

Table 7.7: Comparison of results of cost-matrix with original classification

Modified training set

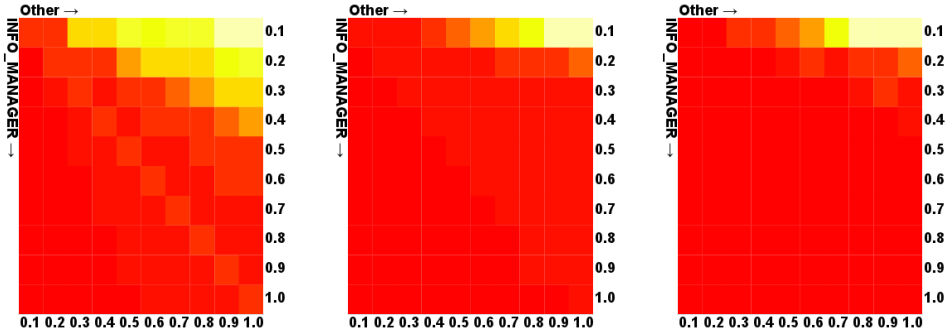
The results of *INFO_MANAGER* recall cost-matrix values have similar distribution like for *full training set*. They are presented in Figure 7.4. Even though they are similar, they are slightly more balanced too. It is caused by the smaller number of examples in the training set when algorithms learned to better distinguish between these two classes. When we look at weighted recall in Figure 7.5 we can see that algorithms had better results

when the weight was set more for the class *INFO_MANAGER*. It is in the contrast with previous Figure 7.4 when the best results were achieved when *OTHER* class had more weight. When we compare the numbers, the best result for the weighted recall was *0.53* but with the same weight of classes *INFO_MANAGER* recall it was *0.99*. This is an important investigation for



(a) J48 (b) JRip (c) SGD

Figure 7.4: Visual analysis of cost-matrix results of IM recall for modified training set



(a) J48 (b) JRip (c) SGD

Figure 7.5: Visual analysis of cost-matrix results of recall for modified training set

debugging of filtering classifier. We mainly want so that no example of *INFO_MANAGER* will be labelled as it belongs to OTHER class. Therefore it is acceptable to let go also some documents belonging to *OTHER* class. But we definitely have to find good ratio to make pre-filtering usable. As

	Cost-Matrix		*Original*	
	Recall	**IM Recall**	**Recall**	**IM Recall**
SGD	0.98	0.99	0.95	0.83
J48	0.94	0.83	0.87	0.83
JRip	0.94	0.87	0.87	0.83

Table 7.8: Comparison of results of cost-matrix with original classification

we can see in Table 7.8 there is also improvement found for setting weights for classes. While in the previous case the *SGD* had only slight improvement, in this case it is more significant. The best achieved result was *0.99* as we mentioned. Also tree-based algorithms had quite better results.

Conclusion

Seeing the result, we can proclaim that setting the weight for particular classes may improve (but also worsen) the results of algorithms. Some algorithms are resistant to setting classes weights and they somehow know to find the correct weight. Nevertheless, also in this case we can apply the rule that decreasing the baseline of the training set leads to better classification of unseen examples.

Chapter 8

Conclusions

"I've always believed that if you put in the work, the results will come."

– Michael Jordan

In this thesis we dealt with the problem of finding people on the Internet. We made a survey of how to search for people through web search engines and what pre-processing actions are essential. Therefore in this thesis we primarily focused on the search of potential business partners and we also discussed differences between *WePS* and other difficulties connected with this problem. We proposed the algorithm of search by hand for business partners which may serve us as the basis for the real application. It is important to search at right places and for this reason we made a survey of suitable information resources.

In the practical part of this work we tried to learn classification algorithms to distinguish between documents about managers working in the field of informatics and other documents. This classification might help to improve the system of searching for business partners in the field of informatics. It can be the first filter of retrieved documents from the Internet since we receive a huge amount of irrelevant data.

We looked at the problem of filtering documents about managers working in the field of informatics (*INFO_MANAGER* class) in two different ways. The first point of view was when we extracted these documents out of all other and tried to learn classifier to distinguish them. The second viewpoint was training classifiers on two different classes of documents: *INFORMATICS* and *MANAGERS*. After that we tested these two classifiers to recognize documents from *INFO_MANAGER* class.

Because of the initial unbalanced distribution of classes we decided to execute two kinds of tests. The first was when we trained classifiers on the original distribution of classes. The second one was when we decreased the baseline of training set to 60%.

The results we got were relatively surprising in a positive way. The results for the training set where the baseline was decreased showed us better performance on unseen data. It means algorithms learned better to recognize documents which we are interested in.

Two stage classification of documents also showed us good performance. Even though we trained documents on different classes, they were able to recognize *INFO_MANAGER* class very well. It is matter of the fact that the results of two stage classification and original classification are comparable.

The results of cost-matrix classification showed that we can improve some algorithms by setting the weight for *INFO_MANAGER* and *OTHER* class. It is possible to control which class we prefer. We can also smooth too high differences between the number of examples.

8.1 Possible improvements

The classification showed us that it is possible to use this method for pre-filtering documents. However, there are several improvements we can consider.

For the two stage classification we can try to compare the result of the classification with some value. For example when *Naive Bayes* returns us probability that the document belongs to some class we can set the threshold value when we accept this classification.

Another improvement may be to find the good baseline value for the training set. We used 60% but this value might not be the best one and we can reach better results by modifying this value.

8.2 Future work

Even though we provided the basic background of how to search for business partners and the pre-filtering of received documents might be very helpful too, there is still so much work to do in this area of informatics.

- **Active learning** one of the possible improvement is active learning when downloaded documents are classified and if the classification is wrong, the user corrects the results.

- **Query refinement** Similarly to human search for people, on the Internet the algorithm is able to search as well. When somebody searches on the Internet he or she uses more refinements of the query to retrieve more relevant documents.

- **Named entities** Extraction of named entities may help with classification of documents and disambiguation of persons.
- **Main content extraction** Extraction and analysis of the main content of documents may improve classification. The noisy content of the document might lead to bad performance of the algorithm.
- **Search on hidden resources** Web search engines usually do not index the whole content of the Internet. Some pages may contain important information for person disambiguation.
- **Intelligent agent system** System can by considered as an agent. At the beginning there exists only one agent for every person. By using the Internet resources and basic knowledge about a person, the agent tries to find and identify himself.

Appendix A

Attachments

Google	*Yahoo!*	*Bing*
getCacheUrl	getClickUrl	getCacheUrl
getContent	getDate	getDateTime
getTitle	**getDescription**	getDeepLinks
getTitleNoFormatting	getDisplayUrl	**getDescription**
getUnescapedUrl	getSize	getDisplayUrl
getUrl	**getTitle**	**getTitle**
getVisibleUrl	**getUrl**	**getUrl**

Bold marked text is used for common API methods.

Table A.1: Overview of API methods for search engine services

Group	Type
Web pages	HTML (.htm, .html, other file extensions)
PDF/PS documents	Adobe Portable Document Format (.pdf) Adobe PostScript (.ps)
MS Office	Microsoft Excel (.xls, .xlsx) Microsoft PowerPoint (.ppt, .pptx) Microsoft Word (.doc, .docx)
Open/Libre Office	OpenOffice presentation (.odp) OpenOffice spreadsheet (.ods) OpenOffice text (.odt)
Text documents	TeX/LaTeX (.tex) Text (.txt, .text, other file extensions) Rich Text Format (.rtf, .wri)
Flash animation	Adobe Flash (.swf)
Source code files	Basic source code (.bas) C/C++ source code (.c, .cc, .cpp, .cxx, .h, .hpp) C# source code (.cs) Java source code (.java) Perl source code (.pl) Python source code (.py)
XML documents	XML (.xml) Google Earth (.kml, .kmz) GPS eXchange Format (.gpx) Wireless Markup Language (.wml, .wap) Scalable Vector Graphics (.svg)
Other documents	Autodesk Design Web Format (.dwf) Hancom Hanword (.hwp)

Table A.2: The most common file types indexed by Google[13]

66.7%	UTF-8
18.0%	ISO-8859-1
3.6%	GB2312
3.4%	Windows-1251
1.8%	Windows-1252
1.8%	Shift JIS
0.9%	GBK
0.9%	Windows-1256
0.6%	ISO-8859-2
0.4%	EUC-JP
0.4%	ISO-8859-15
0.3%	ISO-8859-9
0.3%	Windows-1250
0.2%	Windows-1254
0.2%	Big5
0.2%	EUC-KR
0.1%	Windows-874
0.1%	US-ASCII
0.1%	ISO-8859-7
0.1%	TIS-620
0.1%	Windows-1255
less than 0.1%	Windows-1253, KOI8-R, Windows-1257, UTF-16, ISO-8859-5, KS C 5601, Windows-31J, GB18030, ISO-8859-6, ISO-8859-8, ISO-8859-4, KOI8-U, ISO-2022-JP, ISO-8859-16, ISO-8859-3, ISO-8859-13, Windows-1258, ANSI_X3.110-1983, IBM850, ISO-8859-10, ISO-8859-11, Big5 HKSCS, Windows-949

Table A.3: Usage of encoding [30]

Company	*Person name*
2N Telekomunikace a. s.	Oldřich Stejskal
ABRA Software, a. s.	Ing. Martin Jirmann
	David Rožek
	Ing. Martin Bok
	Ing. Petr Nejedlík
	Ing. Petr Vacek
AG COM, a. s.	Inh. Viktor Horák
	Mgr. Milan Janda
	Gabriel Koós
BERIT, a. s.	Jozef Klein
	Martin Morávek
	Michal Navrátil
	Tomáš Osuský
	David Stoppani
EG - Expert, s. r. o.	Ing. Paška Petr
	Ing. Chrástek Josef
	Ing. Kuřík Stanislav
	Ing. Koukol Stanislav
	Ing. Štiková Hana
	Ing. Šnyta Petr
	Klenčík Martin
	Mgr. Vránová Lidmila
	Kukla Miloslav
IPEX a. s.	Ing. Petr Chruňák
	Ing. Pavel Píštěk
ITeuro a. s.	Petr Boháč
	Jaroslav Bartoš
Kapsch, s. r. o.	Ing. Karel Feix
	Ing. Zbyněk Juřena
NESS Czech s. r. o.	Milan Sameš

Table A.4: The list of collected people

A	Á À Ă Ắ Ằ Ẵ Ả Â Ấ Ầ Ẫ Ẩ Ǎ Å Ǻ Ä Ǟ Ã Ȧ Ǡ Ạ Ā Ả Ȁ Ā Ặ Ậ Ḁ Ⱥ ∀ ɑ	a	á à ă ắ ằ ẵ ả â ấ ầ ẫ ẩ ǎ å ǻ ä ǟ ã ȧ ǡ ạ ā ả ȁ ā ặ ậ ḁ ⱥ ɐ ɑ
B	Ḃ Ḅ Ḇ Ƀ Ɓ	b	ḃ ḅ ḇ ƀ ɓ ƃ ḃ
C	Ć Ĉ Č Ċ Ç Ḉ Ȼ Ƈ	c	ć ĉ č ċ ç ḉ ȼ ƈ ɕ
D	Ď Ḋ Ḍ Ḏ Ḑ Ḓ Đ Ɖ Ɗ ᗞ	d	ď ḋ ḍ ḏ ḑ ḓ đ ɖ ɗ ᵭ ԁ ɗ ᶁ ɗ
E	É È Ĕ Ê Ế Ề Ễ Ể Ě Ë Ẽ Ė Ȩ Ḝ Ę Ē Ḗ Ḕ Ẻ Ȅ Ê Ệ Ȇ Ḝ Ę Ḙ Ḛ Ɇ	e	é è ĕ ê ế ề ễ ể ě ë ẽ ė ȩ ḝ ę ē ḗ ḕ ẻ ȅ ề ê ệ ȇ ḝ ę ḙ ḛ ɇ ᶒ ɇ
F	Ḟ Ƒ	f	ḟ ƒ ᵮ ᶂ
G	Ǵ Ğ Ĝ Ǧ Ġ Ģ Ḡ Ǥ Ɠ	g	ǵ ğ ĝ ǧ ġ ģ ḡ ǥ ɠ ᶃ
H	Ĥ Ȟ Ḧ Ḣ Ḥ Ḫ Ħ Ⱨ	h	ĥ ȟ ḧ ḣ ḥ ḫ ẖ ħ ⱨ
I	Í Ì Ĭ Î Ǐ Ï Ḯ Ĩ Į Ī Ỉ Ȉ Ȋ Ị Ḭ ɨ	i	í ì ĭ î ǐ ï ḯ ĩ į ī ỉ ȉ ȋ ị ḭ ɨ ᵻ ᶖ ɪ
J	Ĵ ɟ	j	ĵ ɟ ǰ ȷ ʝ ɟ ʄ
K	Ḱ Ǩ Ḳ Ķ Ḵ Ƙ Ⱪ	k	ḱ ǩ ḳ ķ ḵ ƙ ⱪ
L	Ĺ Ľ Ḷ Ḹ Ļ Ḽ Ḻ Ł Ŀ Ⱡ Ɬ Ꞁ	l	ĺ ľ ḷ ḹ ļ ḽ ḻ ł ŀ ɫ ɬ ꞁ ɭ ȴ ʟ
M	Ḿ Ṁ Ṃ Ɱ	m	ḿ ṁ ṃ ɱ ᵯ ᶆ
N	Ń Ǹ Ň Ñ Ṅ Ṇ Ṋ Ṉ Ɲ Ꞑ Ŋ ɳ	n	ń ǹ ň ñ ṅ ṇ ṋ ṉ ɲ ꞑ ŋ ɳ ᵰ ᶇ ɳ ȵ
O	Ó Ò Ŏ Ô Ố Ồ Ỗ Ổ Ǒ Ö Ȫ Ő Õ Ṍ Ṏ Ȭ Ȯ Ȱ Ø Ǿ Ǫ Ǭ Ō Ṓ Ṑ Ỏ Ȍ Ȏ Ơ Ớ Ờ Ỡ Ở Ợ Ọ Ộ Ɵ Ↄ	o	ó ò ŏ ô ố ồ ỗ ổ ǒ ö ȫ ő õ ṍ ṏ ȭ ȯ ȱ ø ǿ ǫ ǭ ō ṓ ṑ ỏ ȍ ȏ ơ ớ ờ ỡ ở ợ ọ ộ ɵ ↄ ᴑ
P	Ṕ Ṗ Ƥ Ᵽ	p	ṕ ṗ ƥ ᵽ ᵱ
Q	Ɋ ᑫ	q	ɋ ʠ ɋ
R	Ŕ Ř Ṙ Ŗ Ȓ Ṛ Ṝ Ȑ Ɍ Ʀ	r	ŕ ř ṙ ŗ ȓ ṛ ṝ ȑ ɾ ɍ ɽ ɼ ɾ
S	Ś Š Ŝ Š Ş Ṡ Ș Ṣ Ṩ Ꞩ	s	ś š ŝ š ş ṡ ſ ș ṣ ṩ ꞩ ᵴ ʂ ȿ
T	Ť Ṫ Ţ Ṭ Ț Ṱ Ṯ Ŧ Ⱦ Ƭ Ʈ	t	ť ṫ ţ ṭ ț ṱ ṯ ŧ ⱦ ƫ ƭ ʈ ẗ ⱦ ʇ
U	Ú Ù Ŭ Û Ǔ Ů Ü Ǘ Ǜ Ǚ Ǖ Ű Ũ Ṹ Ų Ū Ṻ Ủ Ȕ Ù Ȗ Ư Ứ Ừ Ữ Ử Ự Ụ Ṳ Ṷ Ṵ Ʉ	u	ú ù ŭ û ǔ ů ü ǘ ǜ ǚ ǖ ű ũ ṹ ų ū ṻ ủ ȕ ù ȗ ư ứ ừ ữ ử ự ụ ṳ ṷ ṵ ʉ ᵾ ᶙ ᵿ
V	Ṽ Ṿ Ʋ	v	ṽ ṿ ʋ ᶌ ⱱ
W	Ẃ Ẁ Ŵ Ẅ Ẇ Ẉ Ⱳ	w	ⱳ ẘ ẃ ẁ ŵ ẅ ẇ ẉ
X	Ẍ Ẋ	x	ẍ ẋ ᶍ
Y	Ý Ỳ Ŷ Ÿ Ỹ Ẏ Ȳ Ỷ Ỿ Ƴ	y	ý ỳ ŷ ẙ ÿ ỹ ẏ ȳ ỷ ỿ ƴ ʏ
Z	Ź Ẑ Ž Ż Ẓ Ẕ Ⱬ Ƶ	z	ź ẑ ž ż ẓ ẕ ⱬ ƶ ᵶ ᶎ ȥ ʐ

Table A.5: The list of supported diacritic

		Recall				IM Recall				Precision				Accuracy				Baseline		
	Algorithm	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Train	Test	Loops
1.	SGD	0.98	0.01	0.96	1	0.71	0.17	0.25	1	0.98	0.01	0.95	1	98.2	0.99	95.71	100	94	94	56
2.	SMO	0.98	0.01	0.95	0.99	0.71	0.15	0.25	1	0.98	0.01	0.94	0.99	98.15	0.98	95	99.29	94	94	56
3.	J48	0.96	0.01	0.93	0.99	0.69	0.18	0.25	1	0.97	0.01	0.93	0.99	96.39	1.43	92.86	99.29	94	94	56
4.	JRip	0.96	0.02	0.91	1	0.67	0.19	0.13	1	0.96	0.02	0.92	1	95.91	1.75	91.43	100	94	94	56
5.	RandomTree	0.96	0.01	0.93	0.99	0.67	0.17	0.25	1	0.96	0.01	0.94	0.99	96.34	1.36	92.86	99.29	94	94	56
6.	Logistic	0.97	0.01	0.94	0.99	0.67	0.16	0.38	1	0.97	0.01	0.93	0.99	96.54	1.1	93.57	99.29	94	94	56
7.	MultiClassClassifier	0.97	0.01	0.94	0.99	0.67	0.16	0.38	1	0.97	0.01	0.93	0.99	96.54	1.1	93.57	99.29	94	94	56
8.	PART	0.97	0.01	0.94	0.99	0.66	0.18	0.38	1	0.97	0.01	0.93	0.99	96.9	1.3	93.57	99.29	94	94	56
9.	VotedPerceptron	0.98	0.01	0.96	1	0.66	0.16	0.25	1	0.98	0.01	0.95	1	97.79	0.96	95.71	100	94	94	56
10.	SimpleLogistic	0.97	0.01	0.95	1	0.65	0.18	0.25	1	0.97	0.01	0.94	1	97.4	1.21	95	100	94	94	56
11.	NaiveBayes	0.98	0.01	0.95	0.99	0.65	0.17	0.25	1	0.98	0.01	0.94	0.99	97.54	0.9	95	99.29	94	94	56
12.	RandomCommittee	0.98	0.01	0.96	1	0.64	0.16	0.25	1	0.98	0.01	0.96	1	97.93	0.94	95.71	100	94	94	56
13.	RandomForest	0.98	0.01	0.96	1	0.63	0.17	0.25	1	0.98	0.01	0.96	1	97.86	0.97	95.71	100	94	94	56
14.	LogitBoost	0.97	0.01	0.95	0.99	0.63	0.18	0.25	0.88	0.97	0.01	0.94	0.99	97.37	1.16	95	99.29	94	94	56
15.	ClassificationViaRegression	0.96	0.02	0.92	0.99	0.57	0.18	0.25	0.88	0.96	0.02	0.92	0.99	96.36	1.66	92.14	99.29	94	94	56
16.	FilteredClassifier	0.96	0.01	0.93	0.99	0.54	0.18	0.13	0.88	0.96	0.02	0.92	0.99	96.47	1.3	92.86	98.57	94	94	56
17.	Bagging	0.97	0.01	0.94	1	0.52	0.19	0.13	1	0.97	0.01	0.93	1	96.77	1.16	94.29	100	94	94	56
18.	REPTree	0.96	0.02	0.92	0.99	0.49	0.26	0	0.88	0.95	0.03	0.89	0.99	95.6	1.52	92.14	98.57	94	94	56
19.	RandomSubSpace	0.97	0.01	0.94	0.99	0.44	0.16	0	0.88	0.96	0.02	0.89	0.99	96.56	1	94.29	99.29	94	94	56
20.	BayesNet	0.96	0.01	0.92	0.99	0.43	0.2	0.13	1	0.95	0.02	0.91	0.99	95.51	1.33	92.14	99.29	94	94	56
21.	AdaBoostM1	0.96	0.01	0.94	0.99	0.42	0.17	0.13	0.88	0.96	0.02	0.91	0.99	96.21	1.24	93.57	99.29	94	94	56
22.	OneR	0.96	0.01	0.94	0.99	0.41	0.14	0.13	0.75	0.96	0.01	0.92	0.99	96.05	0.95	94.29	98.57	94	94	56
23.	LWL	0.96	0.01	0.93	0.99	0.37	0.2	0	0.88	0.96	0.01	0.89	0.99	96.05	1.16	92.86	98.57	94	94	56
24.	LibSVM	0.96	0.01	0.94	0.98	0.26	0.14	0	0.63	0.96	0.01	0.89	0.98	95.75	0.78	94.29	97.86	94	94	56
25.	DecisionStump	0.94	0.01	0.9	0.96	0.07	0.16	0	0.63	0.9	0.02	0.89	0.97	94.4	0.8	90	96.43	94	94	56

Table A.6: Complete results for all classifiers on Managers in the field of Informatics with full training set

		Recall				IM Recall				Precision				Accuracy				Baseline		
	Algorithm	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Train	Test	Loops
1.	SGD	0.95	0.02	0.89	0.99	0.83	0.14	0.5	1	0.97	0.01	0.93	0.99	94.89	2.1	88.57	98.57	60	94	56
2.	JRip	0.87	0.08	0.56	0.97	0.83	0.16	0.38	1	0.95	0.02	0.9	0.98	87.18	7.66	56.43	97.14	60	94	56
3.	J48	0.87	0.05	0.74	0.95	0.83	0.13	0.38	1	0.95	0.01	0.93	0.97	87.09	4.74	73.57	95	60	94	56
4.	MultiClassClassifier	0.9	0.05	0.75	0.98	0.81	0.15	0.5	1	0.95	0.01	0.93	0.98	90.47	5.42	75	97.86	60	94	56
5.	Logistic	0.9	0.05	0.75	0.98	0.81	0.15	0.5	1	0.95	0.01	0.93	0.98	90.47	5.42	75	97.86	60	94	56
6.	SimpleLogistic	0.92	0.03	0.85	0.98	0.8	0.17	0.38	1	0.96	0.01	0.92	0.98	92.13	2.87	85	97.86	60	94	56
7.	FilteredClassifier	0.88	0.05	0.75	0.95	0.8	0.14	0.38	1	0.95	0.01	0.91	0.97	88.04	4.97	75	95	60	94	56
8.	LogitBoost	0.92	0.03	0.83	0.99	0.8	0.17	0.38	1	0.96	0.02	0.92	0.99	92.1	3.27	82.86	98.57	60	94	56
9.	PART	0.88	0.04	0.74	0.94	0.79	0.14	0.38	1	0.95	0.01	0.92	0.97	87.78	4.4	73.57	93.57	60	94	56
10.	VotedPerceptron	0.89	0.05	0.76	0.96	0.79	0.16	0.5	1	0.95	0.01	0.92	0.97	88.79	4.86	76.43	96.43	60	94	56
11.	ClassificationViaRegression	0.87	0.05	0.71	0.96	0.77	0.17	0.38	1	0.95	0.01	0.91	0.97	87.08	5.48	70.71	95.71	60	94	56
12.	SMO	0.94	0.02	0.88	0.97	0.76	0.16	0.38	1	0.96	0.01	0.93	0.98	94.35	2.15	87.86	97.14	60	94	56
13.	RandomTree	0.83	0.07	0.63	0.95	0.76	0.18	0.38	1	0.94	0.01	0.91	0.96	82.78	6.73	62.86	95	60	94	56
14.	AdaBoostM1	0.93	0.04	0.79	0.99	0.75	0.15	0.38	1	0.95	0.01	0.93	0.99	92.58	3.55	79.29	98.57	60	94	56
15.	Bagging	0.92	0.03	0.81	0.98	0.75	0.18	0.38	1	0.95	0.01	0.92	0.98	92.23	3.11	81.43	97.86	60	94	56
16.	RandomCommittee	0.96	0.02	0.9	0.99	0.74	0.17	0.38	1	0.96	0.01	0.94	0.99	95.55	1.69	90	98.57	60	94	56
17.	RandomSubSpace	0.92	0.04	0.76	0.97	0.73	0.18	0.25	1	0.95	0.01	0.91	0.97	91.94	3.82	75.71	97.14	60	94	56
18.	RandomForest	0.95	0.02	0.89	0.98	0.73	0.17	0.38	1	0.96	0.01	0.92	0.98	95.23	1.99	89.29	97.86	60	94	56
19.	REPTree	0.89	0.07	0.6	0.97	0.7	0.22	0.25	1	0.95	0.02	0.9	0.98	89.04	7.23	60	97.14	60	94	56
20.	LWL	0.92	0.07	0.54	0.98	0.7	0.17	0.25	1	0.95	0.02	0.91	0.98	91.85	6.58	54.29	97.86	60	94	56
21.	NaiveBayes	0.97	0.01	0.94	0.99	0.69	0.19	0.38	1	0.97	0.01	0.93	0.99	97.17	1.33	93.57	99.29	60	94	56
22.	OneR	0.93	0.04	0.72	0.98	0.58	0.18	0	1	0.94	0.02	0.89	0.98	92.56	4.11	72.14	97.86	60	94	56
23.	DecisionStump	0.92	0.07	0.44	0.96	0.54	0.21	0	1	0.94	0.02	0.89	0.98	91.9	7.2	43.57	96.43	60	94	56
24.	BayesNet	0.94	0.03	0.81	0.98	0.54	0.25	0.13	1	0.95	0.02	0.91	0.98	94.44	2.67	81.43	97.86	60	94	56
25.	LibSVM	0.96	0.01	0.94	0.98	0.27	0.16	0	0.63	0.95	0.02	0.89	0.98	95.78	1.01	93.57	97.86	60	94	56

Table A.7: Complete results for all classifiers on Managers in the field of Informatics with decreased baseline for training set to 60

		Recall				IM Recall				Precision				Accuracy				Baseline		
	Algorithm	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Train	Test	Loops
1.	NaiveBayes	0.83	0.02	0.79	0.87	0.75	0.01	0.72	0.75	0.92	0.01	0.89	0.95	82.86	1.85	78.82	87.06	86	86	56
2.	SMO	0.95	0.01	0.93	0.97	0.73	0.03	0.67	0.83	0.95	0.01	0.92	0.97	94.71	0.81	92.55	96.86	86	86	56
3.	SimpleLogistic	0.93	0.02	0.89	0.96	0.71	0.08	0.5	0.83	0.93	0.02	0.89	0.96	92.9	1.54	89.02	96.08	86	86	56
4.	LogitBoost	0.93	0.02	0.87	0.95	0.69	0.07	0.47	0.81	0.93	0.02	0.88	0.95	92.68	1.71	87.45	95.29	86	86	56
5.	MultiClassClassifier	0.93	0.02	0.87	0.96	0.68	0.12	0.33	0.83	0.93	0.02	0.87	0.96	92.79	2.15	87.06	96.08	86	86	56
6.	AdaBoostM1	0.92	0.02	0.87	0.95	0.65	0.08	0.31	0.75	0.92	0.02	0.85	0.96	92.5	1.69	86.67	95.29	86	86	56
7.	PART	0.9	0.02	0.85	0.94	0.64	0.11	0.39	0.83	0.9	0.02	0.84	0.94	90.15	2.06	84.71	94.12	86	86	56
8.	JRip	0.9	0.02	0.84	0.95	0.64	0.11	0.25	0.83	0.9	0.02	0.83	0.95	89.8	2.34	83.92	94.9	86	86	56
9.	Logistic	0.92	0.02	0.87	0.96	0.62	0.13	0.33	0.86	0.91	0.02	0.86	0.96	91.54	2.27	87.06	96.06	86	86	56
10.	DecisionStump	0.91	0.02	0.83	0.93	0.62	0.13	0.11	0.67	0.91	0.03	0.8	0.93	91.06	2.05	83.14	92.94	86	86	56
11.	J48	0.9	0.02	0.81	0.94	0.62	0.11	0.28	0.83	0.9	0.02	0.82	0.94	89.63	2.38	80.78	93.73	86	86	56
12.	RandomCommittee	0.94	0.01	0.91	0.96	0.61	0.06	0.42	0.72	0.94	0.01	0.92	0.96	94.01	0.85	91.37	95.69	86	86	56
13.	ClassificationViaRegression	0.9	0.02	0.86	0.94	0.6	0.08	0.47	0.81	0.9	0.02	0.86	0.94	90.3	1.79	86.27	94.12	86	86	56
14.	RandomTree	0.88	0.02	0.82	0.92	0.59	0.12	0.31	0.83	0.88	0.02	0.82	0.92	87.58	2.25	81.57	91.76	86	86	56
15.	FilteredClassifier	0.91	0.02	0.88	0.94	0.58	0.08	0.39	0.78	0.91	0.02	0.86	0.94	91.02	1.52	87.84	94.12	86	86	56
16.	RandomForest	0.93	0.01	0.89	0.96	0.54	0.09	0.28	0.69	0.93	0.02	0.88	0.96	93.1	1.34	88.63	95.69	86	86	56
17.	LWL	0.9	0.02	0.83	0.93	0.51	0.15	0.19	0.67	0.89	0.03	0.8	0.93	89.99	2.36	83.14	93.33	86	86	56
18.	Bagging	0.91	0.02	0.86	0.94	0.47	0.14	0.17	0.67	0.91	0.02	0.83	0.94	91.27	1.8	86.27	94.12	86	86	56
19.	BayesNet	0.87	0.01	0.84	0.91	0.46	0.04	0.39	0.56	0.87	0.01	0.84	0.9	87.35	1.3	84.31	90.59	86	86	56
20.	REPTree	0.9	0.03	0.84	0.94	0.46	0.18	0	0.69	0.89	0.04	0.74	0.94	89.64	2.54	84.31	94.12	86	86	56
21.	RandomSubSpace	0.91	0.02	0.87	0.95	0.45	0.15	0.17	0.78	0.91	0.03	0.84	0.94	90.98	1.97	86.67	94.51	86	86	56
22.	OneR	0.9	0.02	0.85	0.92	0.4	0.17	0.11	0.67	0.89	0.04	0.8	0.93	90.11	2.46	84.71	92.16	86	86	56
23.	LibSVM	0.87	0	0.87	0.88	0.1	0.03	0.08	0.17	0.88	0.02	0.85	0.9	87.2	0.47	86.67	88.24	86	86	56

Table A.8: Complete results for *Two Stage* Informatics-Manager classification with original baseline

		Recall				IM Recall				Precision				Accuracy				Baseline		
	Algorithm	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Train	Test	Loops
1.	SMO	0.92	0.02	0.86	0.95	0.82	0.03	0.75	0.86	0.93	0.01	0.9	0.95	91.6	1.74	86.27	94.9	60	86	56
2.	SimpleLogistic	0.88	0.03	0.76	0.93	0.8	0.06	0.58	0.92	0.91	0.02	0.86	0.94	87.61	2.93	76.47	92.55	60	86	56
3.	J48	0.82	0.04	0.73	0.88	0.78	0.1	0.42	0.94	0.88	0.02	0.82	0.91	81.67	3.52	72.94	87.84	60	86	56
4.	Logistic	0.89	0.04	0.8	0.95	0.77	0.09	0.53	0.89	0.91	0.02	0.85	0.95	89.43	3.76	79.61	94.9	60	86	56
5.	MultiClassClassifier	0.9	0.02	0.85	0.95	0.77	0.1	0.44	0.86	0.91	0.02	0.86	0.95	90.12	2.43	84.71	94.9	60	86	56
6.	LogitBoost	0.85	0.03	0.77	0.9	0.76	0.09	0.5	0.89	0.89	0.02	0.84	0.92	84.82	2.9	77.25	89.8	60	86	56
7.	ClassificationViaRegression	0.82	0.04	0.75	0.91	0.76	0.09	0.56	0.92	0.88	0.02	0.84	0.93	82.28	3.58	74.9	90.98	60	86	56
8.	NaiveBayes	0.8	0.02	0.74	0.84	0.76	0.02	0.69	0.83	0.91	0.02	0.87	0.95	79.79	2.18	74.12	83.53	60	86	56
9.	PART	0.83	0.04	0.75	0.89	0.75	0.11	0.53	0.94	0.88	0.02	0.84	0.93	83.29	3.52	75.29	89.41	60	86	56
10.	FilteredClassifier	0.83	0.03	0.74	0.89	0.74	0.09	0.5	0.94	0.88	0.02	0.84	0.91	83.22	3.23	73.73	89.41	60	86	56
11.	RandomCommittee	0.91	0.02	0.86	0.96	0.72	0.05	0.61	0.81	0.92	0.01	0.88	0.96	91.19	1.92	85.88	95.69	60	86	56
12.	RandomTree	0.78	0.04	0.67	0.87	0.72	0.1	0.44	0.89	0.86	0.02	0.8	0.9	78	4.46	67.06	86.67	60	86	56
13.	RandomForest	0.91	0.02	0.86	0.94	0.72	0.08	0.5	0.86	0.91	0.02	0.87	0.94	90.79	1.89	86.27	94.12	60	86	56
14.	JRip	0.84	0.05	0.64	0.94	0.72	0.12	0.42	0.92	0.88	0.03	0.83	0.93	84.13	5.23	63.53	93.73	60	86	56
15.	Bagging	0.9	0.02	0.82	0.95	0.7	0.08	0.44	0.83	0.9	0.02	0.87	0.94	89.53	2.37	82.35	94.51	60	86	56
16.	AdaBoostM1	0.89	0.03	0.77	0.93	0.7	0.11	0.39	0.89	0.9	0.02	0.83	0.93	88.55	3.14	76.86	93.33	60	86	56
17.	RandomSubSpace	0.89	0.03	0.83	0.93	0.65	0.12	0.22	0.89	0.9	0.02	0.81	0.93	89.01	2.51	83.14	92.94	60	86	56
18.	BayesNet	0.88	0.02	0.82	0.92	0.63	0.06	0.47	0.78	0.89	0.01	0.86	0.92	88.49	2.08	81.96	92.16	60	86	56
19.	REPTree	0.83	0.07	0.6	0.94	0.61	0.15	0.19	0.89	0.87	0.03	0.79	0.94	83.13	6.95	60.39	94.12	60	86	56
20.	OneR	0.9	0.03	0.82	0.94	0.59	0.16	0.19	0.67	0.9	0.04	0.81	0.94	90.11	3.02	81.96	94.12	60	86	56
21.	DecisionStump	0.9	0.03	0.83	0.93	0.52	0.22	0.11	0.67	0.89	0.04	0.81	0.93	89.62	3.04	82.75	93.33	60	86	56
22.	LWL	0.89	0.03	0.83	0.93	0.52	0.19	0.19	0.69	0.88	0.04	0.8	0.93	89.16	2.95	83.14	93.33	60	86	56
23.	LibSVM	0.9	0.01	0.87	0.92	0.37	0.09	0.17	0.47	0.89	0.02	0.85	0.92	89.87	1.21	87.06	91.76	60	86	56

Table A.9: Complete results for *Two Stage* Informatics-Manager classification with decreased baseline to 60

		Recall				IM Recall				Precision				Accuracy				Baseline		
	Algorithm	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Train	Test	Loops
1.	NaiveBayes	0.93	0.02	0.88	0.97	0.84	0.06	0.69	0.91	0.94	0.01	0.9	0.97	93.37	2.28	87.5	97.07	86	86	56
2.	J48	0.85	0.03	0.77	0.92	0.45	0.16	0.14	0.86	0.85	0.03	0.78	0.93	84.96	2.85	76.86	91.76	86	86	56
3.	JRip	0.84	0.04	0.76	0.93	0.4	0.18	0.06	0.72	0.84	0.04	0.74	0.92	83.98	3.56	76.47	92.55	86	86	56
4.	RandomTree	0.83	0.03	0.75	0.93	0.38	0.15	0.14	0.72	0.83	0.03	0.77	0.92	82.77	3.44	74.9	92.55	86	86	56
5.	SimpleLogistic	0.88	0.02	0.81	0.91	0.38	0.14	0.17	0.69	0.86	0.03	0.79	0.91	87.56	2.41	81.18	91.37	86	86	56
6.	PART	0.84	0.03	0.78	0.92	0.38	0.14	0.14	0.69	0.83	0.03	0.78	0.92	83.79	2.95	77.65	91.76	86	86	56
7.	LogitBoost	0.85	0.02	0.8	0.9	0.34	0.13	0.14	0.67	0.84	0.03	0.78	0.89	85.46	2.27	80	90.2	86	86	56
8.	ClassificationViaRegression	0.85	0.02	0.81	0.9	0.34	0.14	0.08	0.75	0.83	0.03	0.77	0.89	84.96	2.31	80.78	89.8	86	86	56
9.	BayesNet	0.85	0.02	0.81	0.9	0.33	0.14	0	0.58	0.83	0.03	0.74	0.89	84.98	1.93	81.18	89.8	86	86	56
10.	MultiClassClassifier	0.88	0.02	0.81	0.92	0.32	0.17	0.06	0.89	0.87	0.03	0.8	0.92	88.22	1.84	80.78	92.16	86	86	56
11.	SMO	0.89	0.01	0.87	0.91	0.31	0.04	0.22	0.39	0.87	0.02	0.84	0.91	88.52	0.85	86.67	90.59	86	86	56
12.	Logistic	0.88	0.02	0.83	0.93	0.3	0.12	0.08	0.64	0.87	0.02	0.81	0.93	87.87	1.63	83.14	93.33	86	86	56
13.	AdaBoostM1	0.86	0.02	0.81	0.9	0.29	0.15	0.03	0.64	0.84	0.04	0.77	0.9	86	2.09	81.18	90.2	86	86	56
14.	RandomCommittee	0.89	0.01	0.87	0.93	0.26	0.09	0.14	0.5	0.88	0.02	0.84	0.93	88.68	1.41	86.67	92.55	86	86	56
15.	FilteredClassifier	0.85	0.02	0.79	0.89	0.21	0.17	0	0.67	0.82	0.04	0.74	0.9	85.43	1.82	79.22	89.02	86	86	56
16.	DecisionStump	0.85	0.03	0.73	0.88	0.2	0.19	0	0.44	0.8	0.05	0.74	0.87	84.87	2.58	73.33	88.24	86	86	56
17.	LWL	0.85	0.03	0.76	0.9	0.2	0.18	0	0.61	0.8	0.05	0.73	0.89	84.66	2.5	76.47	90.2	86	86	56
18.	RandomForest	0.87	0.01	0.85	0.91	0.16	0.08	0	0.42	0.86	0.03	0.74	0.91	87.49	1.14	85.1	90.98	86	86	56
19.	OneR	0.86	0.02	0.78	0.87	0.13	0.11	0	0.42	0.81	0.05	0.74	0.87	85.67	1.54	77.65	87.45	86	86	56
20.	REPTree	0.85	0.02	0.77	0.87	0.07	0.14	0	0.75	0.76	0.04	0.73	0.88	84.94	2.2	77.25	87.06	86	86	56
21.	Bagging	0.86	0.01	0.82	0.89	0.05	0.08	0	0.42	0.79	0.06	0.74	0.88	85.94	0.88	81.57	89.02	86	86	56
22.	RandomSubSpace	0.86	0	0.84	0.87	0.03	0.09	0	0.42	0.76	0.05	0.74	0.88	85.95	0.49	84.31	87.45	86	86	56
23.	LibSVM	0.86	0	0.86	0.86	0	0	0	0	0.74	0	0.74	0.74	85.88	0	85.88	85.88	86	86	56

Table A.10: Complete results for *Two Stage* Manager-Informatics classification with original baseline

		Recall				IM Recall				Precision				Accuracy				Baseline		
	Algorithm	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Mean	σ	Min	Max	Train	Test	Loops
1.	NaiveBayes	0.85	0.06	0.71	0.93	0.9	0.06	0.71	0.97	0.91	0.02	0.87	0.95	84.73	6.09	70.56	93.48	60	86	56
2.	Logistic	0.87	0.04	0.75	0.93	0.81	0.2	0.31	1	0.91	0.03	0.84	0.95	87.28	3.69	75.4	93.28	60	86	56
3.	MultiClassClassifier	0.86	0.05	0.75	0.95	0.78	0.19	0.28	1	0.9	0.04	0.81	0.96	86.03	5.05	74.51	95.29	60	86	56
4.	ClassificationViaRegression	0.74	0.07	0.5	0.84	0.65	0.16	0.28	0.97	0.85	0.03	0.75	0.9	74.31	7.35	49.8	84.31	60	86	56
5.	SMO	0.87	0.03	0.78	0.94	0.65	0.16	0.36	0.89	0.89	0.03	0.82	0.94	87.25	3.1	78.04	94.12	60	86	56
6.	SimpleLogistic	0.8	0.04	0.7	0.87	0.6	0.14	0.28	0.83	0.85	0.03	0.8	0.9	80.46	4.18	70.2	87.45	60	86	56
7.	J48	0.74	0.06	0.58	0.88	0.6	0.14	0.22	0.86	0.83	0.03	0.76	0.9	74	6.11	57.65	87.84	60	86	56
8.	LogitBoost	0.78	0.04	0.67	0.85	0.57	0.16	0.25	0.86	0.84	0.03	0.77	0.9	78.47	4.4	66.67	85.49	60	86	56
9.	RandomTree	0.71	0.06	0.58	0.83	0.57	0.2	0.14	0.92	0.82	0.04	0.72	0.91	70.69	6.03	57.65	83.14	60	86	56
10.	JRip	0.75	0.07	0.46	0.86	0.56	0.23	0.08	0.78	0.83	0.04	0.74	0.89	74.89	6.71	46.27	86.27	60	86	56
11.	BayesNet	0.8	0.06	0.54	0.87	0.55	0.2	0	1	0.84	0.03	0.74	0.89	79.89	6.04	54.12	87.06	60	86	56
12.	AdaBoostM1	0.8	0.05	0.69	0.89	0.55	0.17	0.17	0.78	0.84	0.03	0.77	0.9	80.37	4.59	69.02	88.63	60	86	56
13.	PART	0.74	0.07	0.49	0.85	0.55	0.15	0.17	0.89	0.82	0.03	0.75	0.89	73.54	6.57	48.63	84.71	60	86	56
14.	RandomCommittee	0.87	0.03	0.78	0.93	0.54	0.15	0.28	0.83	0.87	0.03	0.81	0.93	86.71	2.94	78.04	92.55	60	86	56
15.	FilteredClassifier	0.77	0.11	0.29	0.89	0.53	0.18	0.11	0.81	0.83	0.04	0.69	0.89	77.11	10.86	29.41	89.41	60	86	56
16.	LWL	0.78	0.1	0.34	0.88	0.49	0.26	0.08	0.94	0.83	0.05	0.72	0.89	78.11	9.54	34.12	88.24	60	86	56
17.	RandomForest	0.87	0.03	0.81	0.93	0.45	0.16	0.06	0.78	0.86	0.03	0.77	0.92	86.69	2.62	80.78	92.55	60	86	56
18.	OneR	0.79	0.06	0.56	0.87	0.43	0.23	0.08	0.78	0.82	0.05	0.73	0.88	79.22	6.01	56.47	87.06	60	86	56
19.	RandomSubSpace	0.8	0.06	0.61	0.89	0.37	0.19	0	0.78	0.82	0.04	0.74	0.89	80.47	5.56	61.18	89.02	60	86	56
20.	Bagging	0.84	0.04	0.75	0.91	0.35	0.19	0.08	0.78	0.83	0.04	0.75	0.91	83.96	3.71	75.29	91.37	60	86	56
21.	DecisionStump	0.8	0.09	0.31	0.87	0.35	0.22	0	0.97	0.82	0.05	0.74	0.88	79.9	8.51	30.98	87.45	60	86	56
22.	REPTree	0.78	0.07	0.63	0.87	0.34	0.25	0	0.78	0.8	0.05	0.74	0.87	78.02	6.8	63.14	87.45	60	86	56
23.	LibSVM	0.86	0	0.86	0.86	0	0	0	0	0.74	0	0.74	0.74	85.88	0	85.88	85.88	60	86	56

Table A.11: Two Stage Manager-Informatics classification with baseline set to 60

Appendix B

Content of the attached DVD

Folder:

- `database/` contains exported *Postgres* database with all classifications, preprocessed documents and other statistics used by program.
- `documents/` contains presentations made about this topic.
- `master-thesis/` contains the LaTeX source of this thesis including the bibliography in bibtex file, all figures and necessary packages to render the output.
- `Paris` contains all sources in *Java* of the application which is able to fetch and pre-process documents from the Internet. There are also various sub-applications and scripts used for classification. (Do not remove folder `Temp`, there are downloaded files and database references to them!)
- `research/` contains documents with statistics and data used as for writing this thesis as for the research in this area.
- `resources/` contains collections of various data used during the phase of program development.

Bibliography

[1] Top 5 search engines from april 2011 to march 2012 | StatCounter global stats. http://gs.statcounter.com/#search_engine-ww-monthly-201104-201203-bar, April 2012.

[2] 123people.com. 123people.com | free people search, find the public records of everyone. http://www.123people.com/, April 2012.

[3] Javier Artiles, Julio Gonzalo, and Satoshi Sekine. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. http://nlp.uned.es/~javart/docs/weps2007-slides.pdf http://acl.ldc.upenn.edu/W/W07/W07-2012.pdf, 2007.

[4] Pavel BARKHIN, Zoltan Istvan GYONGYI, and Jan PEDERSEN. Link-based spam detection. http://www.patentlens.net/patentlens/patent/US_2006_0095416_A1/en/, 05 2006.

[5] Česká společnost pro systémovou integraci. ČSSI | Česká společnost pro systémovou integraci. http://www.cssi.cz/cssi/, April 2012.

[6] Microsoft Corporation. Rich text format (RTF) specification version 1.9.1. http://www.microsoft.com/downloads/info.aspx?na=41&srcfamilyid=dd422b8d-ff06-4207-b476-6b5396a18a2b&srcdisplaylang=en&u=http%3a%2f%2fdownload.microsoft.com%2fdownload%2f2%2ff%2f5%2f2f599e18-07ee-4ec5-a1e7-f4e6a9423592%2fWord2007RTFSpec9.doc, March 2008.

[7] Dictionary.com. Namesake | define namesake at dictionary.com. http://dictionary.reference.com/browse/namesake, May 2012.

[8] International Organization for Standardization. ISO/IEC 26300:2006 - information technology – open document format for office applications (OpenDocument) v1.0. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43485, November 2006.

[9] The Apache Software Foundation. Apache PDFBox - apache PDFBox - java PDF library. http://pdfbox.apache.org/, October 2011.

[10] Tom Habing. UIUC DLI glossary. http://dli.grainger.uiuc.edu/glossary.htm, November 1998.

[11] Jiawei Han. Mining heterogeneous information networks by exploring the power of links. Discovery Science, pages 13–30, 2009.

[12] Gargoyle Software Inc. Welcome to HtmlUnit. http://htmlunit.sourceforge.net/, August 2011.

[13] Google Inc. What file types can google index? - webmaster tools help. http://www.google.com/support/webmasters/bin/answer.py?hl=en&answer=35287, November 2011.

[14] Intelius, Inc. People search & directory services powered by intelius. http://www.intelius.com/search-faq.php, April 2012.

[15] iSearch. iSearch - social directory - frequently asked questions. http://www.isearch.com/faq.html, April 2012.

[16] Lili Jiang, Wei Shen, and Jianyong Wang. GRAPE: a system for disambiguating and tagging people names in web search. http://dl.acm.org/citation.cfm?id=1772896, 2010.

[17] Daniel A. Keim, Florian Mansmann, and Jim Thomas. Visual analytics: how much visualization and how much analytics? http://doi.acm.org/10.1145/1809400.1809403, May 2010.

[18] Yegor Kozlov, Marc Johnson, Glen Stampoultzis, Rainer Klute, Avik Sengupta, Shawn Laubach, Danny Mui, Jason Height, Amol S Deshmukh, David Fisher, Josh Micich, Maxim Valyanskiy, Jon Svede, and Sergey Vladimirov. Apache POI - the java API for microsoft documents. http://poi.apache.org/, August 2011.

[19] E. Lefever, T. Fayruzov, V. Hoste, and M. De Cock. Clustering web people search results using fuzzy ants. http://www.sciencedirect.com/science/article/pii/S0020025510002215, 2010. Including Special Section on Virtual Agent and Organization Modeling: Theory and Applications.

[20] Bing Liu. Web data mining: exploring hyperlinks, contents, and usage data. Springer, 2007.

[21] Ning LIU, Jun YAN, Benyu ZHANG, Zheng CHEN, and Jian WANG. Identification of similar queries based on overall and partial similarity of time series. http://www.patentlens.net/patentlens/patent/US_2009_0006365_A1/en/, 01 2009.

[22] Ph.D. Mgr. Pavel Rychlý. Ohákování. http://nlp.fi.muni.cz/cz_accent/, November 2011.

[23] Microsoft. Download: Word 2007: Rich text format (RTF) specification, version 1.9.1 - microsoft download center - download details. http://www.microsoft.com/download/en/details.aspx?id=10725, March 2008.

[24] Ministry of Finance of the CR. ARES - access to registers of economic entities. http://wwwinfo.mfcr.cz/ares/ares.html.en, April 2012.

[25] MyLife.com, Inc. Free people search - find people - wink. http://wink.com/, April 2012.

[26] OpenOffice.org. XML project. http://xml.openoffice.org/, November 2011.

[27] Karl Pearson. The Grammar of Science. 1900.

[28] PeekYou LLC. PeekYou LLC | indexing the web around people. http://www.peekyou.biz/, April 2012.

[29] pipl. Pipl - people search. http://pipl.com/, April 2012.

[30] Q-Success. Usage statistics of character encodings for websites, november 2011. http://w3techs.com/technologies/overview/character_encoding/all, November 2011.

[31] Bc. Matej Sabo. User:matej.sabo - vyhľadávanie informácií. http://vi.ikt.ui.sav.sk/User:matej.sabo?view=home, December 2010.

[32] Satya Nadella - Senior Vice President. Bing - exciting news from bing and yahoo! - search blog - site blogs - bing community. http://www.bing.com/community/site_blogs/b/search/archive/2010/08/24/exciting-news-from-bing-and-yahoo.aspx, August 2010.

[33] Peter M. Senge. Sharing knowledge. http://www.solonline.org/res/kr/shareknow.html, June 1998.

[34] Leslie Sheh. Daily dose of my mind. http://shehlovee.tumblr.com/post/2100214807, December 2010.

[35] Daniel Turing and Daniel Cassidy. swfmill swf2xml and xml2swf. http://swfmill.org/, November 2011.

[36] Inc. Unicode. Unicode 3.0, chapter 1 - introduction. http://unicode.org/book/uc20ch1.html, October 2005.

[37] Inc. Unicode. FAQ - UTF-8, UTF-16, UTF-32 & BOM. http://unicode.org/faq/utf_bom.html, March 2011.

[38] vnikic. HtmlCleaner project home page. http://htmlcleaner.sourceforge.net/, February 2011.

[39] Quang Minh Vu, Tomonari Masada, Atsuhiro Takasu, and Jun Adachi. Disambiguation of People in Web Search Using a Knowledge Base. 2007 IEEE International Conference on Research, Innovation and Vision for the Future, pages 185–191, March 2007.

[40] w3schools. HTML entities. http://www.w3schools.com/html/html_entities.asp, May 2012.

[41] Eric WATSON, Marcelo CALBUCCI, Sally SALAS, and Darren SHAKIB. Query preprocessing and pipelining. http://www.patentlens.net/patentlens/patent/US_7472113/en/, 12 2008.

[42] WebKnox. Named entity definition ÂŤ WebKnox blog. http://webknox.com/blog/2010/09/named-entity-definition/, September 2010.

[43] Yahoo! Yahoo! search - people search. http://people.yahoo.com/, April 2012.

[44] Minoru Yoshida, M Ikeda, S Ono, and I Sato. Person name disambiguation by bootstrapping. http://dl.acm.org/citation.cfm?id=1835454, 2010.